Data Mining Using Neural Networks: A Guide for Statisticians

Basilio de Bragança Pereira

UFRJ - Universidade Federal do Rio de Janeiro

Calyampudi Radhakrishna Rao

PSU - Penn State University

June 2009

Contents

List of Figures

Chapter 1

Introduction

This work is a tentative to bring the attention of statisticians the parallel work that has been done by neural network researchers in the area of data analysis.

In this area of application (data analysis) it is important to show that most neural network models are similar or identical to popular statistical techniques such as generalized linear models, classification, cluster, projection ..., generalized additive models, principal components, factor analysis etc. On the other hand, neural networks algorithms implementations are inefficient because:

1. They are based on biological or engineering criteria such as how easy is to fit a net on a chip rather than on well-established statistical or optimization criteria;

2. Neural networks are designed to be implemented on massively parallel computers such as PC. On a serial computer, standard statistical optimization criteria usually will be more efficient to neural networks implementation.

In an computational process we have the following hierarchical framework. The top of the hierarchy is the computational level. This attempts to answer the questions - what is being computed and why? The next level is the algorithm, which describes how the computation is being carried out, and finally there is the implementation level, which gives the details and steps of the facilities of the algorithm.

Research in neural networks involve different group of scientists in neurosciences, psychology, engineering, computer science and mathematics. All these groups are asking different questions: neuro-scientists and psychologists want to know how animal brain works, engineers and computer scientists want to build intelligent machines and mathematicians want to understand the fundamentals properties of networks as complex systems.

Mathematical research in neural networks is not specially concerned with statistics but mostly with non-linear dynamics, geometry, probability and other areas of mathematics. On the other hand, neural network research can stimulate statistics not only by pointing to new applications but also providing it with new type of non-linear modeling. We believe that neural network will become one of the standard technique in applied statistics because of its inspiration, but also statisticians have a range of problems in which they can contribute in the neural network research.

Therefore, in this work we will concentrate on the top of the hierarchy of the computational process, that is what is being done and why, and also how the neural network models and statistical models are related to tackle the data analysis real problem.

The book is organized as follows: some basics on artificial neural networks (ANN) is presented in Chapters 2 and 3. Multivariate statistical topics are presented in Chapter 4. Chapter 5 deals with parametric and nonparametric regression models and Chapter 6 deals with inference results, survival analysis, control charts and time series. The references and the bibliography is referred to sections of the chapters. To be fair authors which inspired some of our developments, we have included them in the bibliography, even if they were not explicitly mentioned in the text. The first author is grateful to CAPES, an Agency of the Brazilian Ministry of Education for a onde year support grant to visit Penn State University and to UFRJ for a leave of absence.

Chapter 2

Fundamental Concepts on Neural Networks

2.1 Artificial Intelligence: Symbolist & Connectionist

Here we comment on the connection of neural network and artificial intelligence. We can think of artificial intelligence as intelligence behavior embodied in human-made machines. The concept of what is intelligence is outside the scope of this guide. It comprises the following components: learning, reasoning, problem-solving, perception and language understanding.

From the early days of computing, there have existed two different approaches to the problem of developing machines that might embody such behavior. One of these tries to capture knowledge as a set of irreducible semantic objects or symbols and to manipulate these according to a set of formal rules. The rules taken together form a recipe or algorithm for processing the symbols. This approach is the symbolic paradigm, which can be described as consisting of three phrases:

- choice of an intelligent activity to study;
- development of a logic-symbolic structure able to imitate (such as "if condition 1 and condition 2... then result";)
- compare the efficiency of this structure with the real intelligent activity.

Note that the symbolic artificial intelligence is more concern in imitate intelligence and not in explaining it. Concurrent with this, has been another line of research, which has used machines whose architecture is loosely based on the human brain. These artificial neural networks are supposed to learn from examples and their “knowledge” is stored in representations that are distributed across of a set of weights.

The connectionist, or neural network approach, starts from the premise that intuitive knowledge cannot be captured in a set of formalized rules. It postulated that the physical structure of the brain is fundamental to emulate the brain function and the understanding of the mind. For the connectionists, the mental process comes as the aggregate result of the behavior of a great number of simple computational connected elements (neurons) that exchange signals of cooperation and competition.

The form how these elements are interconnected are fundamental for the resulting mental process, and this fact is that named this approach *connectionist*.

Some features of this approach are as follows:

- information processing is not centralized and sequential but parallel and spacially distributed;

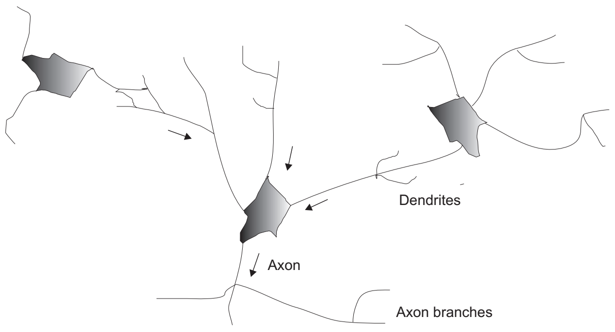
- information is stored in the connections and adapted as new information arrives;

- it is robust to failure of some of its computational elements.

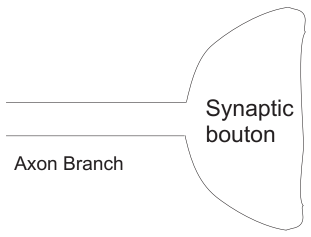
2.2 The brain and neural networks

The research work on artificial neural networks has been inspired by our knowledge of biological nervous system, in particular the brain. The brain has features, we will not be concerned with large scales features such as brain lobes, neither with lower levels of descriptions than nerve cell, call neurons. Neurons have irregular forms, as illustrated in Figure 2.1.

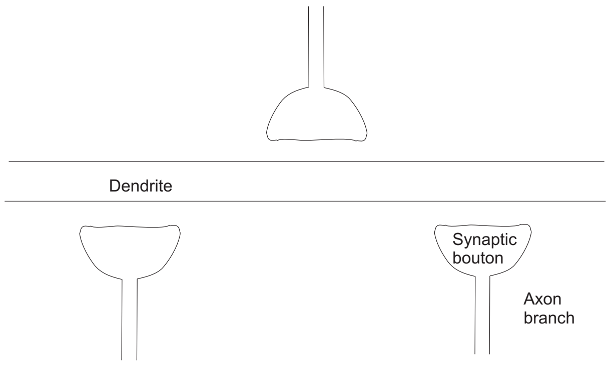
The cell body (the central region) is called neuron. We will focus on morphological characteristic which allow the neuron to function as an information processing device. This characteristic lies in the set of fibers that emanate from the cell body. One of these fibers - the axon - is responsible for transmitting information to other neurons. All other are dendrites, when carry information transmitted from other neurons. Dendrites are surrounded by the synaptic bouton of other neurons, as in Figure 1c. Neurons are highly interconnected. Physical details are given in Table 1 where we can see that brain is a very complicated system, and a true model of the brain would be very complicated. Building such a model is the task that scientists faces. Statisticians as engineers on the contrary, use simplified models that they can actually build and manipulate usefully. As statisticians we will take the view that brain models are used as inspiration to building artificial neural networks as the wings of a bird were the inspiration for the wings of an airplane.



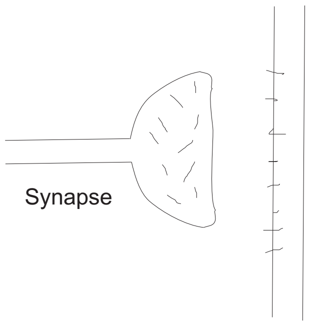
(a) Neurons



(b) Sypatic bouton or synapse



(c) Synaptic bouton making contact with a dendrite



(d) Transmitters and receptor in the synapse and dendrites

Figure 2.1: Neural network elements

Table 2.1: Some physical characteristics of the brain

Number of neurons	100 billion
Number of synapses/neurons	1000
Total number of synapses (Number of processors of a computer)	100000 billion (100 millions)
Operation (Processing speed in a computer)	100 (1 billion)
Number of operations	10000 trillion/s
Human brain volume	300 cm^3
Dendrite length of a neuron	1 cm
Firing length of an axon	10 cm
Brain weight	1.5kg
Neuron weight	1.2×10^{-9}g
Synapse	Excitatory and Inhibitory

2.3 Artificial Neural Networks and Diagrams

An artificial neuron can be identified by three basic elements.

- A set of synapses, each characterized by a weight, that when positive means that the synapse is excitated and when negative is inhibitory. A signal x_j in the input of synapse j connected to the neuron k is multiplied by a synaptic weight w_{jk}.
- Summation - to sum the input signs, weighted by respective synapses (weights).
- Activation function - restrict the amplitude of the neuron output.

The Figure 2.2 presents the model of an artificial neuron:

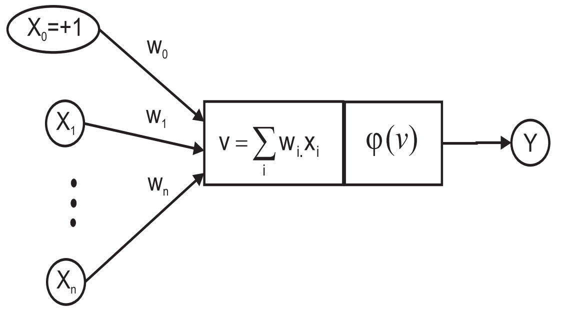


Figure 2.2: Model of an artificial neuron

Models will be displayed as networks diagrams. Neurons are represented by circles and boxes, while the connections between neurons are shown as arrows:

- Circles represent observed variables, with the name shown inside the circle.
- Boxes represent values computed as a function of one or more arguments. The symbol inside the box indicates the type of function.
- Arrows indicate that the source of the arrow is an argument of the function computed at the destination of the arrow. Each arrow usually has a weight or parameter to be estimated.
- A thick line indicates that the values at each end are to be fitted by same criterion such as least squares, maximum likelihood etc.
- An artificial neural network is formed by several artificial neurons, which are inspired in biological neurons.
- Each artificial neuron is constituted by one or more inputs and one output. These inputs can be outputs (or not) of other neurons and the output can be the input to other neurons. The inputs are multiplied by weights and summed with one constant; this total then goes through the activation function. This function has the objective of activate or inhibit the next neuron.
- Mathematically, we describe the k^{th} neuron with the following form:

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \text{ and } y_k = \varphi(u_k - w_{k0}), \tag{2.3.1}$$

where $x_0, x_1, \ldots, x_p$ are the inputs; $w_{k1}, \ldots, w_{kp}$ are the synaptic weights; u_k is the output of the linear combination; w_{k0} is the bias; $\varphi(\cdot)$ is the activation function and y_k is the neuron output.

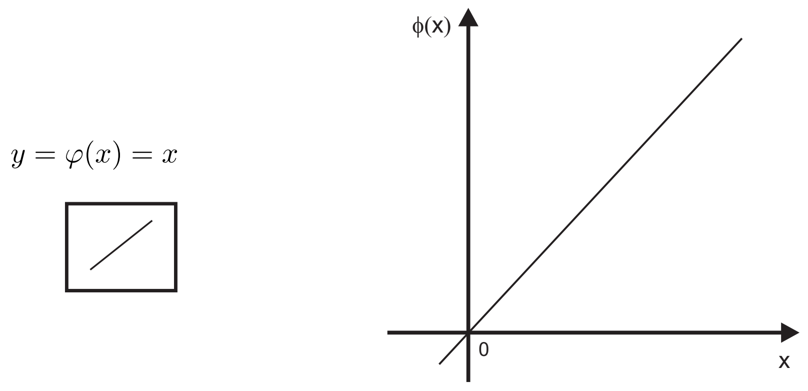
The first layer of an artificial neural network, called *input layer*, is constituted by the inputs x_p and the last layer is the *output layer*. The intermediary layers are called *hidden layers*.

The number of layers and the quantity of neurons in each one of them is determined by the nature of the problem.

A vector of values presented once to all the output units of an artificial neural network is called *case, example, pattern, sample*, etc.

2.4 Activation Functions

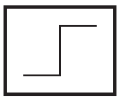
An activation function for a neuron can be of several types. The most commons are presented in Figure 2.3:

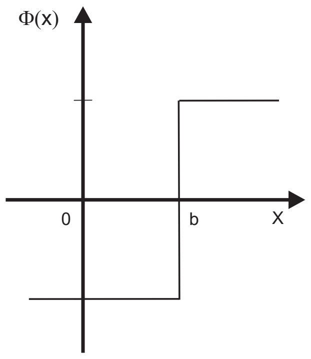


a) Identity function (ramp)

Figure 2.3: Activation functions: graphs and ANN representation

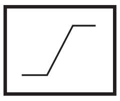
$$y = \varphi(x) = \begin{cases} 1 & x \geq b \\ -1 & x < b \end{cases}$$

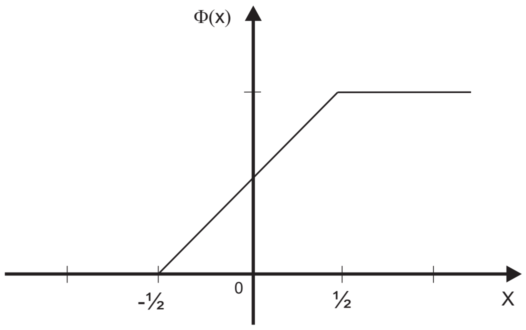




b) Step or threshold function

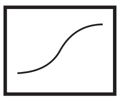
$$y = \varphi(x) = \begin{cases} 1 & x \geq 1/2 \\ z & -1/2 \leq x < \dfrac{1}{2} \\ 0 & x \leq -\dfrac{1}{2} \end{cases}$$

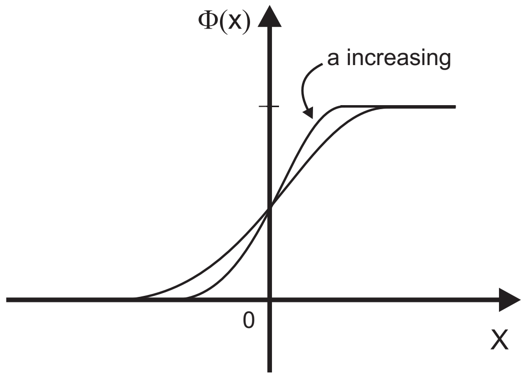




c) Piecewise-linear function

$$y = \varphi_1(x) = \frac{1}{1 + \exp(-ax)}$$

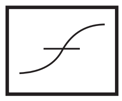


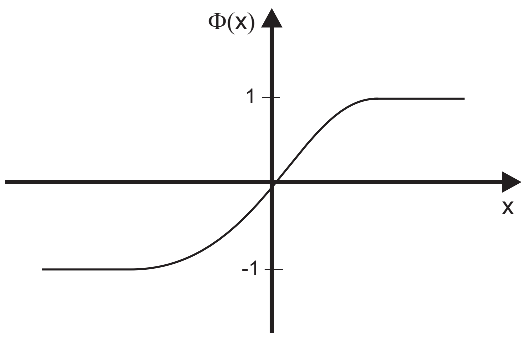


d) Logistic (sigmoid)

Figure 2.3: Activation functions: graphs and ANN representation (cont.)

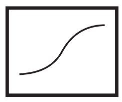
$$y = \varphi_2(x) = 2\varphi_1(x) - 1$$
$$= \frac{1 - \exp(-ax)}{1 + \exp(-ax)}$$





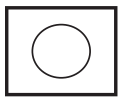
e) Symmetrical sigmoid

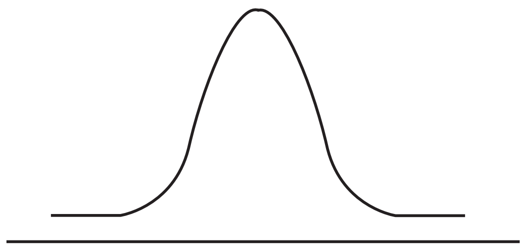
$$y = \varphi_3(x) = tang\left(\frac{x}{2}\right) = \varphi_2(x)$$



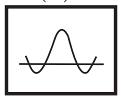
f) Hyperbolic tangent function

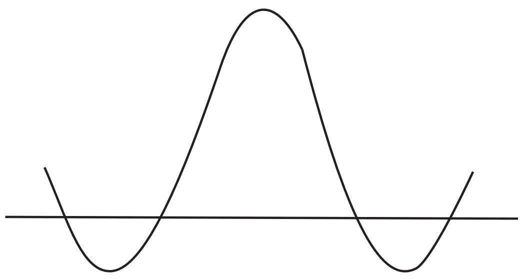
i. Normal $\varphi(x) = \Phi(x)$





ii. Wavelet $\varphi(x) = \Psi(x)$





g) Radial Basis (some examples)

Figure 2.3: Activation functions: graphs and ANN representation (cont.)

h) Wavelet Transform

$\varphi(x) = \sum\sum 2^{j/2}\Psi\left(2^j x - k\right)$

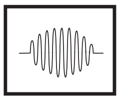


Figure 2.3: Activation functions: graphs and ANN representation (cont.)

The derivatives of the identity function is 1. The derivative of the step function is not defined, which is why it is not used.

The nice feature of the sigmoids is that they are easy to compute. The derivative of the logistic is $\varphi_1(x)(1 - \varphi_1(x))$.

For the hyperbolic tangent the derivative is

$$1 - \varphi_3^2(x). \tag{2.4.1}$$

2.5 Network Architectures

Architecture refers to the manner in which neurons in a neural network are organized. There are three different classes of network architectures.

1. **Single-Layer Feedforward Networks**

   In a layered neural network the neurons are organized in layers. One input layer of *source nodes* projects onto an *output layer* of neurons, but not vice-versa. This network is strictly of a feedforward type.

2. **Multilayer Feedforward Networks**

   These are networks with one or more *hidden layers*. Neurons in each of the layer have as their inputs the output of the neurons of preceding layer only.

   This type of architecture covers most of the statistical applications

   The neural network is said to be fully connected if every neuron in each layer is connected to each node in the adjacent forward layer, otherwise it is partially connected.

3. **Recurrent Network**

   Is a network where at least one neuron connects with one neuron of the preceding layer and creating a feedback loop.

   This type of architecture is mostly used in optimization problems.

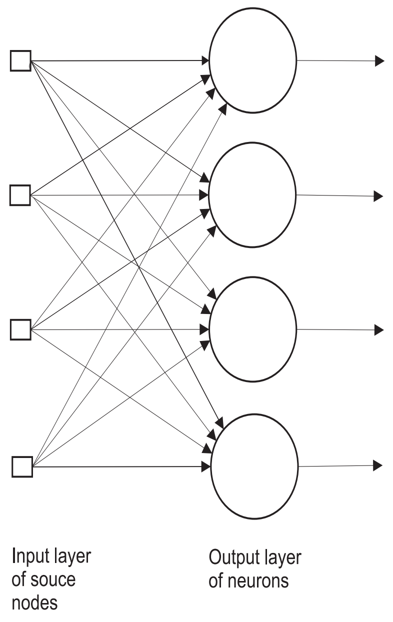


Figure 2.4: Single-layer network

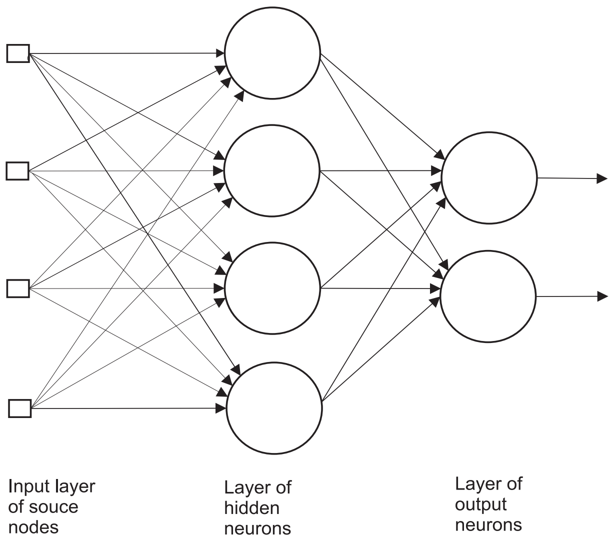


Figure 2.5: Multilayer network

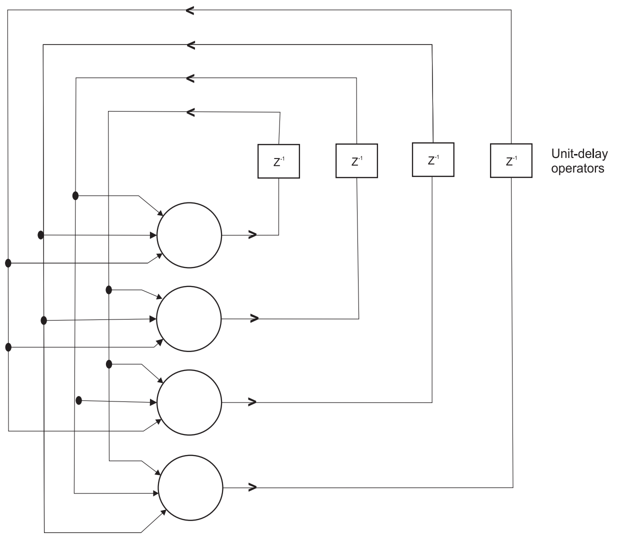


Figure 2.6: Recurrent network

2.6 Network Training

The training of the network consists of adjusting the synapses or weight with the objective of optimizing the performance of the network. From a statistical point of view, the training corresponds to estimating the parameters of the model given a set of data and an estimation criteria.

The training can be as follows:

- learning with a teacher or supervised learning: is the case in which to each input vector the output is known, so we can modify the network synaptic weights in an orderly way to achieve the desired objective;

- learning without a teacher: this case is subdivided into self-organized (or unsupervised) and reinforced learning. In both cases, we do not have the output, which corresponds to each input vector. Unsupervised training corresponds, for example, to the statistical methods of cluster analysis and principal components. Reinforced learning training corresponds to, for example, the credit-assign problem. It is the problem of assigning a credit or a blame for overall outcomes to each of the internal decisions made by a learning machine. It occurs, for example, when the error is propagated back to the neurons in the hidden layer of a feedforward network of section **??**.

2.7 Kolmogorov Theorem

Most conventional multivariate data analysis procedures can be modeled as a mapping

$$f : A \to B$$

where A and B are finite sets. Consider the following examples (Murtagh, 1994):

(i) B is a set of discrete categories, and A a set of m-dimensional vectors characterizing n observations. The mapping f is a clustering function and choosing the appropriate f is the domain of cluster analysis.

(ii) In dimensionality-reduction technique (PCA, FA, etc.), B is a space containing n points, which has lower dimension than A. In these cases, the problem of the data analyst is to find the mapping f, without precise knowledge of B.

(iii) When precise knowledge of B is available, B has lower dimension than A, and f is to be determined, includes the methods under the names discrimination analysis and regression.

Most of the neural network algorithm can also be described as method to determine a nonlinear f, where $f : A \to B$ where A is a set of n, m-dimensional descriptive vectors and where some members of B are known a priori. Also, some of the most common neural networks are said to perform similar tasks of statistical methods. For example, multilayer perception and regression analysis, Kohonen network and multidimensional scaling, ART network (Adaptive Resonance Theory) and cluster analysis.

Therefore the determination of f is of utmost importance and made possible due to a theorem of Andrei Kolmogorov about representation and approximation of continuous functions, whose history and development we present shortly.

In 1900, Professor David Hilbert at the International Congress of Mathematicians in Paris, formulated 23 problems from the discussion of which "advancements of science may be expected". Some of these problems have been related to applications, where insights into the existence, but not the construction, of solutions to problems have been deduced.

The thirteenth problem of Hilbert is having some impact in the area of neural networks, since its solution in 1957 by Kolmogorov. A modern variant of Kolmogorov's Theorem is given in Rojas (1996, p.265)(or Bose and Liang (1996, p.158).

Theorem 1. *(Kolmogorov Theorem): "Any continuous function $f(x_1, x_2, \ldots, x_m)$ on n variables $x_1, x_2, \ldots, x_m$ on $I_n = [0,1]^m$ can be represent in the form*

$$f(x_1, \ldots, x_m) = \sum_{q=1}^{2m+1} g\left(\sum_{p=1}^{m} \lambda_p h_q(x_p)\right)$$

where g and h_q, for $q = 1, \ldots, 2m+1$ are functions of one variable and λ_p for $p = 1, \ldots, m$ are constants, and the g_q functions do not depend on $f(x_1, \ldots, x_m)$.

Kolmogorov Theorem states an exact representation for a continuous function, but does not indicate how to obtain those functions. However if we want to approximate a function, we do not demand exact reproducibility but only a bounded approximate error. This is the approach taken in neural network.

From Kolmogorov's work, several authors have improved the representation to allow Sigmoids, Fourier, Radial Basis, Wavelets functions to be used in the approximation.

Some of the adaptations of Kolmogorov to neural networks are (Mitchell, 1997, p.105):

- **Boolean functions**. Every boolean function can be represented exactly by some network with two layers of units, although the number of hidden units required grows exponentially in the worst case with the number of network inputs.

- **Continuous functions**. Every bounded continuous function can be approximated with arbitrary small error by a network with two layers. The result in this case applies to networks that use sigmoid activation at the hidden layer and (unthresholded) linear activation at the output layer. The number of hidden units required depends on the function to be approximated.

- **Arbitrary functions**. Any function can be approximated to arbitrary accuracy by a network with three layers of units. Again, the output layer uses linear activation, the two hidden layers use sigmoid activation, and the number of units required in each hidden layer also depends on the function to be approximated.

2.8 Model Choice

We have seen that the data analyst's interest is to find or learn a function f from A to B. In some cases (dimension reduction, cluster analysis), we don't have precise knowledge of B and the term "unsupervised" learning is used. On the other hand, the term "supervised" is often applied to the situation where f is to be determined, but where some knowledge of B is available (e.g. discriminant and regression analysis).

In the case of supervised learning the algorithm to be used in the determination of f embraces the phases: learning, where the mapping f is determined using the training set, $B' \subset B$; testing of how well f performs using a test set $B'' \subset B$, B'' not identical to B'; and an application. The testing and application phases are termed generalization.

In a multilayer network, Kolmogorov's results provide guidelines for the number of layers to be used. However, some problems may be easier to solve using two hidden layers. The number of units in each hidden layer and (the number) of training interactions to estimate the weights of the network have also to be specified. In this section, we will address each of these problems.

2.8.1 Generalization

By generalization we mean that the input-output mapping computed by the network is correct for a test data never used. It is assumed that the training is a representative sample from the environment where the network will be applied.

Given a large network, it is possible that loading data into the network it may end up memorizing the training data and finding a characteristic (noise, for example) that is presented in the training data but not true in the underline function that is to be modelled.

The training of a neural network may be view as a curve fitting problem and if we keep improving our fitting, we end up with a very good fit only for the training data and the neural network will not be able to generalize (interpolate or predict other sample values).

2.8.2 Bias-variance trade-off: Early stopping method of training

Consider the two functions in Figure 2.7 obtained from two training of a network.

The data was generated from function $h(x)$, and the data from

$$y(x) = h(x) + \epsilon. \tag{2.8.1}$$

The training data are shown as circles ($\circ$) and the new data by (+). The dashed represents a simple model that does not do a good job in fitting the new data. We say that the model has a bias. The full line represents a more complex model

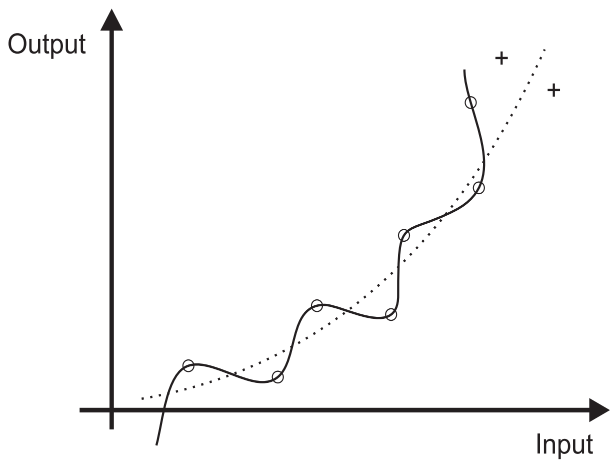


Figure 2.7: Network fits: - high variance , . . . high bias

which does an excellent job in fitting the data, as the error is close to zero. However it would not do a good job for predicting the values of the test data (or new values). We say that the model has high variance. We say in this case that the network has memorize the data or is overfitting the data.

A simple procedure to avoid overfitting is to divide the training data in two sets: estimation set and validation set. Every now and then, stop training and test the network performance in the validation set, with no weight update during this test. Network performance in the training sample will improve as long as the network is learning the underlying structure of the data (that is $h(s)$). Once the network stops learning things that are expected to be true of any data sample and learns things that are true only of the training data (that is ϵ), performance on the validation set will stop improving and will get worse.

Figure 2.8 shows the errors on the training and validation sets. To stop overfitting we stop training at epoch E_0.

Haykin (1999, p.215) suggest to take 80% of the training set for estimation and 20% for validation.

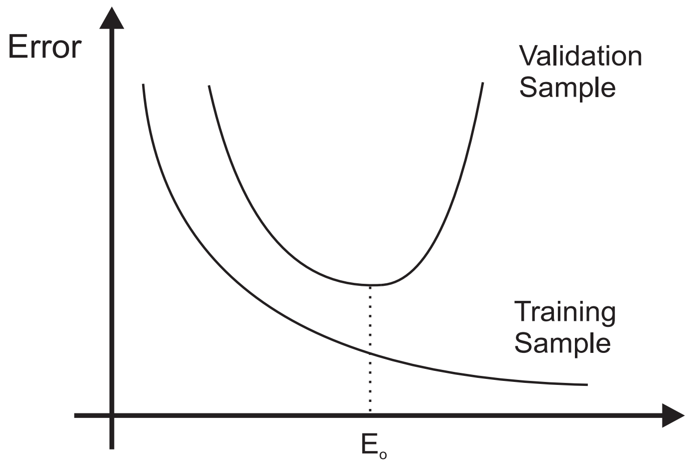


Figure 2.8: Model choice: training x validation

2.8.3 Choice of structure

A large number of layers and neurons reduce bias but at the same time increase variance. In practice it is usual to try several types of neural networks and to use cross-validation on the behavior in a test data, to choose the simplest network with a good fit.

We may achieve this objective in one of two ways.

- Network growing
- Network pruning

2.8.4 Network growing

One possible policy for the choice of the number of layers in a multilayer network is to consider a nested sequence of networks of increasing sequence as suggest by Haykin (1999, p.215).

- p networks with a single layer of increasing size $h'_1 < h'_2 < \ldots < h'_p$.
- g networks with two hidden layers: the first layer of size h_p and the second layer is of increasing size $h''_1 < h''_2 < \ldots < h''_g$.

The cross-validation approach similar to the early stopping method previously presented can be used for the choice of the numbers of layers and also of neurons.

2.8.5 Network pruning

In designing a multilayer perception, we are building a nonlinear model of the phenomenon responsible for the generation of the input-output examples used to train the network. As in statistics, we need a measure of the fit between the model (network) and the observed data.

Therefore the performance measure should include a criterion for selecting the model structure (number of parameter of the network). Various identification or selection criteria are described in the statistical literature. The names of Akaike, Hannan, Rissanen, Schwartz are associated with these methods, which share a common form of composition:

$$\begin{pmatrix}\text{Model-Complexity}\\ \text{criteria}\end{pmatrix} = \begin{pmatrix}\text{Performance}\\ \text{measure}\end{pmatrix} + \begin{pmatrix}\text{Model-Complexity}\\ \text{penalty}\end{pmatrix}$$

The basic difference between the various criteria lies in the model-complexity penalty term.

In the context of back-propagation learning or other supervised learning procedure the learning objective is to find a weight vector that minimizes the total risk

$$R(W) = E_s(W) + \lambda E_c(W) \tag{2.8.2}$$

where $E_s(W)$ is a standard performance measure which depends on both the network and the input data. In back-propagation learning it is usually defined as mean-square error. $E_c(W)$ is a complexity penalty which depends on the model alone.

The regularization parameter λ, which represents the relative importance of the complexity term with respect to the performance measure. With $\lambda = 0$, the network is completely determined by the training sample. When λ is large, it says that the training sample is unreliable.

Note that this risk function, with the mean square error is closely related to ridge regression in statistics and the work of Tikhonov on ill posed problems or regularization theory (Orr, 1996, Haykin, 1999, p.219).

A complexity penalty used in neural networks is (Haykin, 1999, p.220)

$$E\{w\} = \sum_i \frac{(w_i/w_o)^2}{1 + (w_i/w_o)^2} \tag{2.8.3}$$

where w_0 is a preassigned parameter. Using this criteria, weight's w_i's with small values should be eliminated.

Another approach to eliminate connections in the network, works directly in the elimination of units and was proposed and applied by Park et al (1994). It consists in applying PCA, principal components analysis as follows:

(i) initially train a network with an arbitrary large number P of hidden modes;

(ii) apply PCA to the covariance matrix of the outputs of the hidden layer;

(iii) choose the number of important components P^* (by one of the usual criterion: eigenvalues greater than one, proportion of variance explained etc.);

(iv) if $P^* < P$, pick P^* nodes out of P by examining the correlation between the hidden modes and the selected principal components.

Note that the resulting network will not always have the optimal number of units. The network may improve its performance with few more units.

2.9 Terminology

Although many neural networks are similar to statistical models, a different terminology has been developed in neurocomputing. Table 2.2 helps to translate from the neural network language to statistics (cf Sarle, 1996). Some further glossary list are given in the references for this section. Some are quite specialized character: for the statistician, e.g.: Gurney (1997) for neuroscientists term, Addrians and Zantinge (1996) for data mining terminology.

Table 2.2: Terminology

Neural Network Jargon	**Statistical Jargon**
Generalizing from noisy data and assessment of accuracy thereof	Statistical Inference
The set of all cases one wants to be able to generalize to	Population
A function of the value in a population, such as the mean or a globally optimal synaptic weight	Parameter

Continued on next page

Neural Network Jargon	**Statistical Jargon**
A function of the values in a sample, such as the mean or a learned synaptic weight	Statistic
Neuron, neurocode, unit, node processing element	A simple linear or nonlinear computing element that accepts one or more inputs, computes a function thereof, and may direct the result to one or more other neurons
Neural Networks	A class of flexible nonlinear regression and discriminant models, data reduction models, and nonlinear dynamical systems consisting of an often large number of neurons interconnected in often complex ways and often organized in layers
Statistical methods	Linear regression and discriminant analysis, simulated annealing, random search
Architecture	Model
Training, Learning, Adaptation	Estimation, Model fitting, Optimization
Classification	Discriminant analysis
Mapping, Function approximation	Regression
Supervised learning	Regression, Discriminant analysis
Unsupervised learning, Self-organization	Principal components, Cluster analysis, Data reduction
Competitive learning	Cluster analysis
Hebbian learning, Cottrell/ Munzo/ Zisper technique	Principal components
Training set	Sample, Construction sample
Test set, Validation set	Hold-out sample

Continued on next page

Neural Network Jargon	**Statistical Jargon**
Pattern, Vector, Example, Case	Observation, Case
Binnary (0/1), Bivalent or Bipolar (-1/1)	Binary, Dichotomous
Input	Independent variables, Predictors, Regressors, Explanatory variables, Carries
Output	Predicted values
Forward propagation	Prediction
Training values	Dependent variables, Responses
Target values	Observed values
Training par	Observation containing both inputs and target values
Shift register, (Tapped) (time) delay (line), Input window	Lagged value
Errors	Residuals
Noise	Error term
Generalization	Interpolation, Extrapolation, Prediction
Error bars	Confidence intervals
Prediction	Forecasting
Adaline (ADAptive LInear NEuron)	Linear two-group discriminant analysis (not a Fisher´s but generic)
(No-hidden-layer) perceptron	Generalized linear model (GLIM)
Activation function, Signal function, Transfer function	Inverse link function in GLIM
Softmax	Multiple logistic function

Continued on next page

Neural Network Jargon	**Statistical Jargon**
Squashing function	Bounded function with infinitive domain
Semilinear function	Differentiable nondecreasing function
Phi-machine	Linear model
Linear 1-hidden-layer perceptron	Maximum redundant analysis, Principal components of instrumental variables
1-hidden-layer perceptron	Projection pursuit regression
Weights Synaptic weights	(Regression coefficients, Parameter estimates)
Bias	Intercept
The difference between the expected value of a statistic and the corresponding true value (parameter)	Bias
OLS (Orthogonal least square)	Forward stepwise regression
Probabilistic neural network	Kernel discriminant analysis
General regression neural network	Kernel regression
Topolically distributed enconding	(Generalized) Additive model
Adaptive vector quantization	Iterative algorithm of doubtful convergence for K-means cluster analysis
Adaptive Resonance Theory 2^nd	Hartigan´s leader algorithm
Learning vector quantization	A form of piecewise linear discriminant analysis using a preliminary cluster analysis
Counterpropagation	Regressogram based on k-means clusters

Continued on next page

Neural Network Jargon	**Statistical Jargon**
Encoding, Autoassociation	Dimensionality reduction (Independent and dependent variables are the same)
Heteroassociation	Regression, Discriminant analysis (Independent and dependent variables are different)
Epoch	Iteration
Continuous training, Incremental training, On-line training, Instantaneous training	Iteratively updating estimates one observation at time via difference equation, as in stochastic approximation
Batch training, Off-line training	Iteratively updating estimates after each complete pass over the data as in most nonlinear regression algorithms
Shortcuts, Jumpers, Bypass connections, direct linear feedthrough (direct connections from input to output)	Main effects
Funcional links	Iteration terms or transformations
Second-order network	Quadratic regression, Response-surface model
Higher-order networks	Polynomial regression, linear model with iteration terms
Instar, Outstar	Iterative algorithms of doubtful convergence for approximating an arithmetic mean or centroid
Delta-rule, adaline rule, Widrow-Hoff rule, LMS (Least Mean Square) rule	Iterative algorithm of doubtful convergence for training a linear perceptron by least squares, similar to stochastic approximation
Training by minimizing the median of the squared errors	LMS (Least Median of Square)

Continued on next page

Neural Network Jargon	**Statistical Jargon**
Generalized delta rule	Iterative algorithm of doubtful convergence for training a non-linear perceptron by least squares, similar to stochastic approximation
Backpropagation	Computation of derivatives for a multilayer perceptron and various algorithms (such as generalized delta rule) based thereon
Weight decay, Regularization	Shrinkage estimation, Ridge regression
Jitter	Random noise added to the inputs to smooth the estimates
Growing, Prunning, Brain damage, self-structuring, ontogeny	Subset selection, Model selection, Pre-test estimation
Optimal brain surgeon	Wald test
LMS (Least Mean Squares)	OLS (Ordinary Least Squares) (see also ”LMS rule” above)
Relative entropy, Cross entropy	Kullback-Leibler divergence
Evidence framework	Empirical Bayes estimation

2.10 Mcculloch-pitt Neuron

The McCulloch-Pitt neuron is a computational unit of binary threshold, also called Linear Threshold Unit (LTU). The neuron receive the weighted sum of inputs and has as output the value 1 if the sum is bigger than this threshold and it is shown in Figure 2.9.

Mathematically, we represent this model as

$$y_k = \varphi\left(\sum_{j=1}^{p} \omega_{kj} - \omega_o\right) \tag{2.10.1}$$

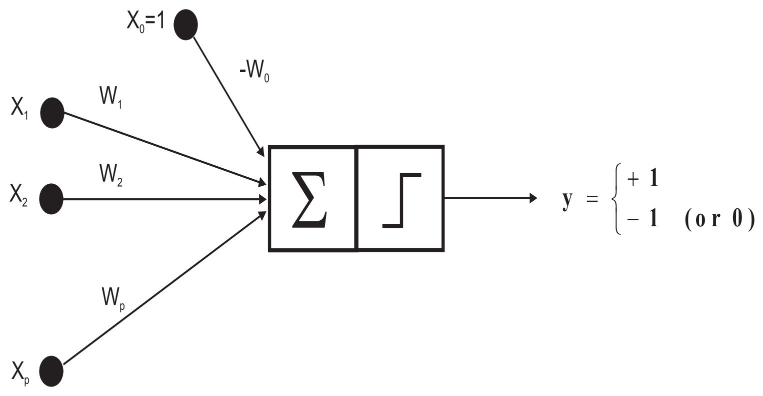


Figure 2.9: McCullogPitt neuron

where y_k is an output of neuron k, ω_{jk} is the weight of neuron j to neuron k. x_j is the output of neuron j, ω_o is the threshold to the neuron k and φ is the activation function defined as

$$\varphi(\text{input}) = \begin{cases} 1 & \text{if input} \geq \omega_0 \\ -1 & \text{otherwise} \end{cases} \tag{2.10.2}$$

2.11 Rosenblatt Perceptron

The Rosenblatt perceptron is a model with several McCulloch-Pitt neurons. Figure 2.10 shows one such a model with two layers of neurons

The one layer perceptron is equivalent to a linear discriminant:

$$\sum \omega_j x_j - \omega_0. \tag{2.11.1}$$

This perceptron creates an output indicating if it belongs to class 1 or -1 (or 0). That is

$$y = \text{Out} = \begin{cases} 1 & \sum \omega_i x_i - \omega_0 \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.11.2}$$

The constant ω_0 is referred as threshold or bias. In the perceptron, the weights are constraints and the variables are the inputs. Since the objective is to estimate (or

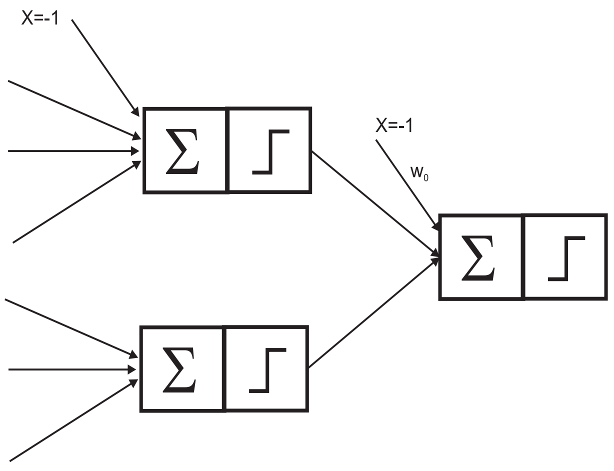


Figure 2.10: Two layer perceptron

learn) the optimum weights, as in other linear discriminants, the next step is to train the perceptron.

The perceptron is trained with samples using a sequential learning procedure to determine the weight as follows: The samples are presented sequentially, for each wrong output the weights are adjusted to correct the error. If the output is corrected the weights are not adjusted. The equations for the procedure are:

$$\omega_i(t+1) = \omega_i(t) + \Delta\omega_i(t) \tag{2.11.3}$$
$$\omega_0(t+1) = \omega_0(t) + \Delta\omega_0(t) \tag{2.11.4}$$

and

$$\Delta\omega_i(t) = (y - \hat{y})x_i \tag{2.11.5}$$
$$\Delta\omega_0(t) = (y - \hat{y}) \tag{2.11.6}$$

Formally, the actual weight in true t is $\omega_i(t)$, the new weight is $\omega_i(t+1)$, $\Delta\omega_i(t)$ is the adjustment or correction factor, y is the correct value or true response, $\hat{y}$ is the perceptron output, a is the activation value for the output $\hat{y} = f(a)$ ($a = \sum \omega_i x_i$).

The initial weight values are generally random numbers in the interval 0 and 1. The sequential presentation of the samples continues indefinitely until some stop criterion is satisfied. For example, if 100 cases are presented and if there is no error, the training will be completed. Nevertheless, if there is an error, all 100 cases are presented again to the perceptron. Each cycle of presentation is called an *epoch*. To make learning and convergence faster, some modifications can be done in the algorithm. For example,

(i) To normalize the data: all input variables must be within the interval 0 and 1.

(ii) To introduce a *learning rate*, α, in the weight actualization procedure to $\alpha\omega_i$. The value of α is a number in the interval 0 and 1.

The perceptron converge theorem states that when two classes are linearly separable, the training guarantees the converge but not how fast. Thus if a linear discriminant exists that can separate the classes without committing an error, the training procedure will find the line or separator plan, but it is not known how long it will take to find it (Figure 2.11).

Table 2.3 illustrates the computation of results of this section for the example of Figure 2.11. Here convergence is achieved in one epoch. Here $\alpha = 0.25$.

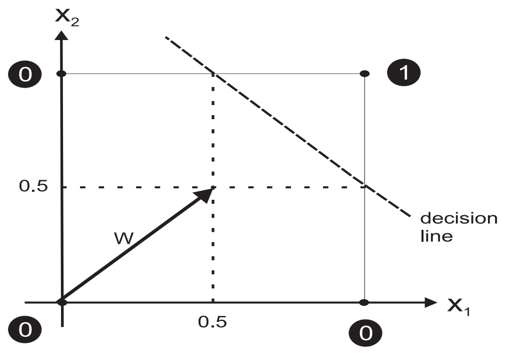


Figure 2.11: Two-dimensional example

Table 2.3: Training with the perceptron rule on a two-input example

ω_1	ω_2	ω_0	x_1	x_2	a	y	$\hat{y}$	$a(\hat{y}-y)$	$\Delta\omega_1$	$\Delta\omega_2$	$\Delta\omega_0$
0.0	0.4	0.3	0	0	0	0	0	0	0	0	0
0.0	0.4	0.3	0	1	0.4	1	0	-0.25	0	-0.25	0.25
0.0	0.15	0.55	1	0	0	0	0	0	0	0	0
0.0	0.15	0.55	1	1	0.15	0	1	0.25	0.25	0.25	-0.25

2.12 Widrow's Adaline and Madaline

Another neuron model, using as a training method the method of Least Mean Square (LMS) error, was developed by Widrow and Hoff and called ADALINE (ADAptive LINear Element). A generalization which consist of a structure of many ADALINE is the MADALINE.

The major difference between the Perceptron (or LTU) and the ADALINE is in the activation function. While the perceptron uses the step or threshold function, the ADALINE uses the identity function as in Figure 2.12.

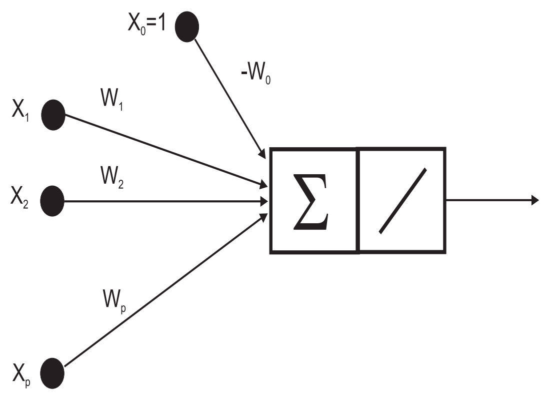


Figure 2.12: ADALINE

To a set of samples, the performance measure fo the training is:

$$D_{MLS} = \sum (\tilde{y} - y)^2 \tag{2.12.1}$$

The LMS training objective is to feed the weigths that minimize D_{MLS}.

A comparison of training for the ADALINE and Perceptron for a two dimensional neuron with $w_0 = -1$, $w_1 = 0.5$, $w_2 = 0.3$ and $x_1 = 2$, $x_2 = 1$ would result in an activation value of:

$$a = 2x0.5 + 1x3 - 1 = 0.3 \tag{2.12.2}$$

Table 2.4 gives the updating results for the sample case.

The technique used to update the weight is called gradient descent. The perceptron and the LMS search for a linear separator. The appropriate line or plane can only be obtained if the class are linear separable. To overcome this limitation the more general neural network model of the next chapter can be used.

Table 2.4: Comparison of updating weights

	Perceptron	Adaline
$\tilde{y}$	1	0.3
$w_1(t+1)$	$0.5 + (0-1).2 = -1.5$	$0.5 + (0-0.3) = -0.1$
$w_2(t+1)$	$0.3 + (0-1).2 = -0.7$	$0.3 + (0-0.3).1 = 0$
$w_0(t+1)$	$-1 + (0-1) = -2$	$-1 + (0-0.3) = -1.3$

Chapter 3

Some Common Neural Networks Models

3.1 Multilayer Feedforward Networks

This section deals with networks that use more than one perceptron arranged in layers. This model is able to solve problems that are not linear separable such as in Figure 3.1 (it is of no help to use layers of ADALINE because the resulting output would be linear).

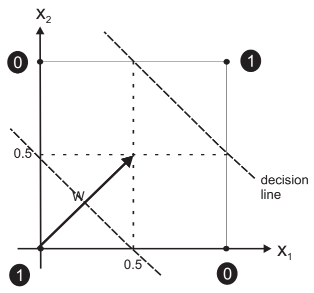
In order to train a new learning rule is needed. The original perceptron learning cannot be extended to the multilayered network because the output function is not differentiable. If a weight is to be adjusted anywhere in the network, its effect in the output of the network has to be known and hence the error. For this, the derivative of the error or criteria function with respect to that weight must be found, as used in the delta rule for the ADALINE.

So a new activation function is needed that is non-linear, otherwise non-linearly separable functions could not be implemented, but which is differentiable. The one that is most often used successfully in multilayered network is the sigmoid function.

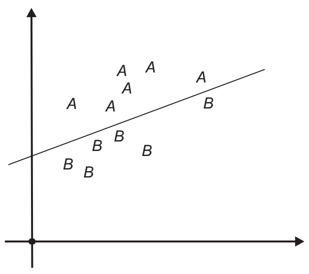
Before we give the general formulation of the generalized delta rule, called backpropagation, we present a numerical illustration for the Exclusive OR (XOR) problem of Figure 3.1(a) with the following inputs and initial weights, using the network of Figure 3.2.

The neural network of Figure 3.2 has one hidden layer with 2 neurons, one input layer with 4 samples and one neuron in the output layer.

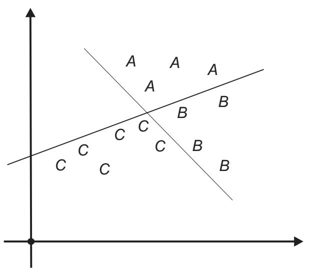
Table 3.1 presents the forward computation for one sample (1,1) in the XOR problem.



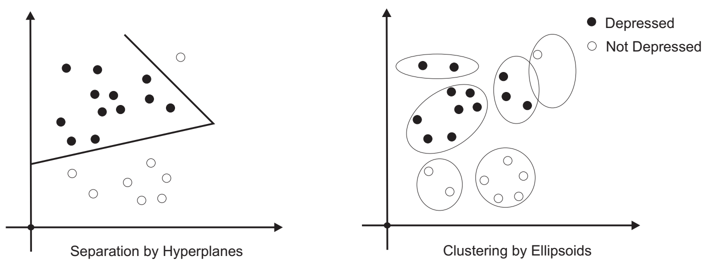
(a) Exclusive OR (XOR)



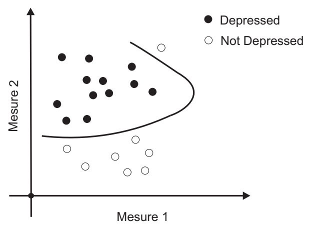
(b) Linear separable class



(c) Non-linear separable



(d) Separable by two-layer of perceptron



(e) Separable by feedforward (sigmoid) network, psychological measures

Figure 3.1: Classes of problems

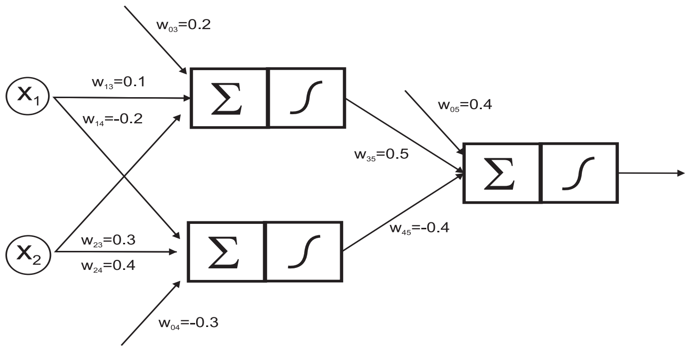


Figure 3.2: Feedforward network

Table 3.1: Forward computation

Neuron	Input	Output
3	.1+.3+.2+.6	$1/(1+\exp(-.6)) = .65$
4	.-2.2+.4-.3-.1	$1/(1+\exp(.1)) = .48$
5	$.5^*.65 - .4^*.48 + .4 = .53$	$1/(1+\exp(-.53)) = .63$

Table 3.2 gives the calculus for the backpropagation of the error for the XOR example.

Table 3.2: Error backpropagation

Neuron	Error	Adjust of tendency
5	$.63^*(1-.63)^*(0-.63) = -.147$	-.147
4	$.48^*(1-.48)^*(-.147)^*(-.4) = .015$	.015
3	$.65^*(1-.65)^*(-.147)^*(-.4) = -.017$	-.017

Table 3.3 gives the weight adjustment for this step of the algorithm.

Table 3.3: Weight adjustment

Weight	Value
w_{45}	$-.4 + (-.147)^*.48 = -.47$
w_{35}	$.5 + (-.147)^*.65 = .40$
w_{24}	$.4 + (.015)^*1 = .42$
w_{23}	$.3 + (-017)^*1 = .28$
w_{14}	$-.2 + (.015)^*1 = .19$
w_{13}	$.1 + (-.017)^*1 = .08$
w_{05}	$.4 + (-.147) = .25$
w_{04}	$-.3 + (.015) = -.29$
w_{03}	$.2 + (-.017) = .18$

The performance of this update procedure suffer the same drawbacks of any gradient descent algorithm. Several heuristic rules have been suggested to improve its performance, in the neural network literature (such as normalize the inputs to be in the interval 0 and 1, initial weights chosen at random in the interval

-0.5 and 0,5; to include a momentum parameter to speed convergence etc.) Here we only present the general formulation of the algorithm.

The backpropagation algorithm

Let L be the number of layers. The different layers are denoted by $\mathcal{L}_1$ (the input later), $\mathcal{L}_2, \dots, \mathcal{L}_M$ (the output layer). We assume the output has M neurons. When neurons i and j are connected a weight ω_{ij} is associated with the connection. In a multilayer feedforward network, only neurons in subsequent layers can be connected:

$$\omega_{ij} = 0 \Rightarrow i\epsilon\mathcal{L}_{l+1},\ j \in \mathcal{L}_l\ 1 \le l \le L-1.$$

The state or input or neuron j is characterized by z_j. The network operates as follows. The input layer assigns z_j to neurons j in $\mathcal{L}_1$. The output of the neurons in $\mathcal{L}_2$ are

$$z_k = f\left(\sum_{j\in\mathcal{L}_1} \omega_{kj} z_j\right)\ k \in \mathcal{L}_2 \tag{3.1.1}$$

where f is an activation function that has derivatives for all values of the argument. For a neuron m in an arbitrary layer we abbreviate

$$a_m = \sum_{j\in\mathcal{L}_{l-1}} \omega_{mj} z_j\ 2 \le l \le L \tag{3.1.2}$$

and where the outputs of layer $l-1$ is known,

$$z_m = f(a_m)\ m \in \mathcal{L}_l. \tag{3.1.3}$$

We concentrate on the choice of weights when one sample (or pattern) is presented to the network. The desired output, true value, or target will be denoted by $t_k, k \in \mathcal{L}_L\ (k = 1, \dots, M)$.

We want to adapt the initial guess for the weights so that to decrease:

$$D_{LMS} = \sum_{k\in\mathcal{L}_L} \left(t_k - f(a_k)\right)^2. \tag{3.1.4}$$

If the function f is nonlinear, then D is a nonlinear function of the weights, the gradient descent algorithm for the multilayer network is as follows.

The weights are updated proportional to the gradient of D with respect to the weights, the weights ω_{mj} is changed by an amount

$$\Delta\omega_{mj} = -\alpha\frac{\partial D}{\partial \omega_{mj}}\ m \in \mathcal{L}_l,\ j \in \mathcal{L}_{l-1} \tag{3.1.5}$$

where α is called the *learning rate*. For the last and next to last layer the updates are given by substituting (3.1.3) and (3.1.4) in (3.1.5).

$$\begin{aligned}\Delta\omega_{mj} &= -\alpha\frac{\partial}{\partial\omega_{mj}}\sum_{p\in\mathcal{L}_L}\left[t_p - f\left(\sum_{q\in\mathcal{L}_{L-1}}\omega_{pq}z_q\right)\right]^2 \\ &= -\alpha 2\sum_{p\in\mathcal{L}_L}\left(t_p - f(a_p)\right)\left(-f'(a_p)\right)\sum_{g\in\%mcl_{L-1}}\delta_{pm}\delta_{gj}z_g \\ &= -\alpha 2\left(t_m - f(a_m)\right)f'(a_m)z_j\, m\in\mathcal{L}_L, j\in\mathcal{L}_{L-1}.\end{aligned} \tag{3.1.6}$$

We assume that the neuron is able to calculate the function f and its derivative f'. The error $2\alpha\left(t_m - f(a_m)\right)f'(a_m)$ can be sent back (or propagated back) to neuron $j\in\mathcal{L}_{L-1}$. The value z_j is present in neuron j so that the neuron can calculate $\Delta\omega_{mj}$. In an analogous way it is possible to derive the updates of the weights ending in $\mathcal{L}_{L-1}$:

$$\begin{aligned}\Delta\omega_{mj} &= -\alpha\frac{\partial}{\partial\omega_{mj}}\sum_{p\in\mathcal{L}_L}\left\{t_p - f\left[\sum_{g\in\mathcal{L}_{L-1}}\omega_{pg}f(\omega_{qr}z_r)\right]\right\}^2 \\ &= -\alpha 2\sum_{p\in\mathcal{L}_L}\left[t_p - f(a_p)\right]\left(-f'(a_p)\right)\sum_{g\in\%mcl_{L-1}}\delta_{gm}\delta_{xj}z_r \\ &= -\alpha 2\sum_{p\in\mathcal{L}_L}\left(t_p - f(a_p)\right)f'(a_p)\omega_{pm}f'(a_m)z_j, m\in\mathcal{L}_{L-1}, j\in\mathcal{L}_{L-2}.\end{aligned} \tag{3.1.7}$$

Now, $2\alpha\left[t_p - f(a_p)\right]f'(a_p)\omega_{pm}$ is the weighted sum of the errors sent from layer $\mathcal{L}_L$ to neuron m in layer $\mathcal{L}_{L-1}$. Neuron m calculate this quantity using $\omega_{pm}, p\in\mathcal{L}_p$. Multiply this quantity $f'(a_m)$ and send it to neuron j that can calculate $\Delta\omega_{mj}$ and update ω_{mj}.

This weight update is done for each layer in decreasing order of the layers, until $\Delta_{mj}, m\in\mathcal{L}_2, j\in\mathcal{L}_1$. This is the familiar backpropagation algorithm. Note that for the sigmoid (symmetric or logistic) and hyperbolic activation function the properties of its derivatives makes it easier to implement the backpropagation algorithm.

3.2 Associative and Hopfield Networks

In unsupervised learning we have the following groups of learning rules:

- competitive learning

• Hebbian learning

When the competitive learning is used, the neurons compete among them to update the weights. These algorithms have been used in classification, signal extraction, cluster. Examples of networks that use these rules are ART (Adaptive Resonance Theory) and SOM (Self Organized Maps) networks.

Hebbian learning is based on the weight actualization due to neurophisiologist Hebb. This algorithm have been used in characteristics extraction, associative memories. Example of network that use these rules are: Hopfield and PCA (Principal Component Analysis) networks. The neural networks we have discussed so far have all been trained in a supervised manner. In this section, we start with associative networks, introducing simple rules that allow unsupervised learning. Despite the simplicity of their rules, they form the foundation for the more powerful networks of latter chapters.

The function of an associative memory is to recognize previous learned input vectors, even in the case where some noise is added. We can distinguished three kinds of associative networks:

- *Heteroassociative networks:* map m input vectors $\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^m$ in p-dimensional spaces to m output vector $\boldsymbol{y}^1, \boldsymbol{y}^2, \dots, \boldsymbol{y}^m$ in k-dimensional space, so that $\boldsymbol{x}^i \rightarrow \boldsymbol{y}^i$.
- *Autoassociative networks* are a special subsets of heteroassociative networks, in which each vector is associated with itself, i.e. $\boldsymbol{y}^i = \boldsymbol{x}^i$ for $i = 1, \dots, n$. The function of the network is to correct noisy input vector.
- *Pattern recognition* (cluster) networks are a special type of heteroassociative networks. Each vector is associated with the scalar (class) i. The goal of the network is to identify the class of the input pattern.

Figure 3.3 summarizes these three networks.

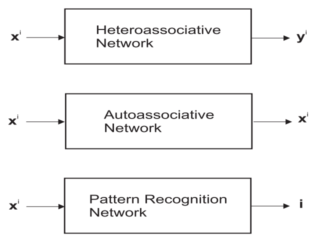


Figure 3.3: Type of associative networks

The three kinds of associative networks can be implemented with a single layer of neurons. Figure 3.4 shows the structure of the heteroassociative network without feedback.

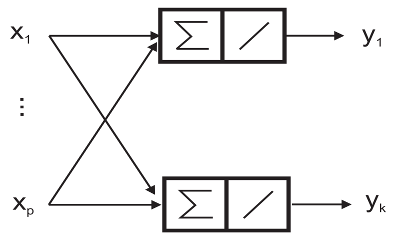


Figure 3.4: Associative network

Let $\boldsymbol{W} = (\omega_{ij})$ the $p \times k$ weight matrix. The row vector $\boldsymbol{x} = (x_1, \ldots, x_p)$ and the identity activation produces the activation

$$\boldsymbol{y}_{1\times k} = \boldsymbol{x}_{1p}\boldsymbol{W}_{p\times k}. \tag{3.2.1}$$

The basic rule to obtain the elements of W that memorize a set of n associations $(\boldsymbol{x}^i\boldsymbol{y}^i)$ is

$$\boldsymbol{W}_{p\times k} = \sum_{i=1}^{n} \boldsymbol{x}^{i'}_{p\times 1} \circ \boldsymbol{y}^{i}_{1k} \tag{3.2.2}$$

that is for n observations, let X be the $n \times p$ matrix whose rows are the input vector and Y be the $n \times k$ matrix whose rows are the output vector. Then

$$\boldsymbol{W}_{p\times k} = \boldsymbol{X}'_{p\times n} \circ \boldsymbol{Y}_{nk}. \tag{3.2.3}$$

Therefore in matrix notation we also have

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W} \tag{3.2.4}$$

Although without a biological appeal, one solution to obtain the weight matrix is the OLAM - Optimal Linear Associative Memory. If $n = p$, X is square matrix, if it is invertible, the solution of (3.2.1) is

$$\boldsymbol{W} = \boldsymbol{X}^{-1}\boldsymbol{Y} \tag{3.2.5}$$

If $n < p$ and the input vectors are linear independent the optimal solution is

$$\boldsymbol{W}_{p\times k} = \left(\boldsymbol{X}'_{pn}\boldsymbol{X}_{np}\right)^{-1}\boldsymbol{X}'_{pn}\boldsymbol{Y} \tag{3.2.6}$$

which minimizes $\sum ||\boldsymbol{W}\boldsymbol{X} - \boldsymbol{Y}||^2$ and $(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ is the pseudo-inverse of X. If the input vectors are linear dependent we have to use regularization theory (similar to ridge regression).

The biological appeal to update the weight is to use the Hebb rule

$$\boldsymbol{W} = \eta\boldsymbol{X}'\boldsymbol{Y} \tag{3.2.7}$$

as each sample presentation where η is a learning parameter in 0 and 1.

The association networks are useful for face, figure etc. recognition and compression, and also function optimization (such as Hopfield network).

We will present some of this association networks using some examples of the literature.

Consider the example (Abi et al, 1999) where we want to store a set of schematic faces given in Figure 3.5 in an associative memory using Hebbian learning. Each face is represented by a 4-dimensional binary vector, denoted by $\boldsymbol{x}^i$ in which a given element corresponds to one of the features from Figure 3.6.

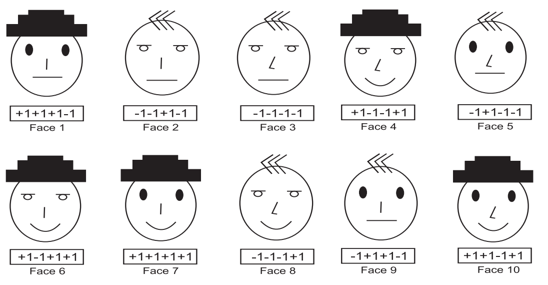


Figure 3.5: Schematic faces for Hebbian learning. Each face is made of four features (hair,eyes,nose and mouth), taking the value $+1$ or -1

Suppose that we start with and $\eta = 1$.

$$\boldsymbol{W}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.2.8}$$

The first face is store in the memory by modifying the weights to

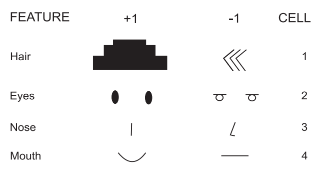


Figure 3.6: The features used to build the faces in Figure 3.5

$$\boldsymbol{W}_1 = \boldsymbol{W}_0 + \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \end{bmatrix}, \tag{3.2.9}$$

the second face is store by

$$\boldsymbol{W}_2 = \boldsymbol{W}_1 + \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & -2 \\ 0 & 0 & -2 & +2 \end{bmatrix}, \tag{3.2.10}$$

and so on until

$$\boldsymbol{W}_{10} = \boldsymbol{W}_9 + \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 2 & 2 & 6 \\ 2 & -10 & 2 & -2 \\ 2 & 2 & 10 & -2 \\ 6 & -2 & -2 & 10 \end{bmatrix}. \tag{3.2.11}$$

Another important association network is the Hopfield network. In its presentation we follow closely Chester (1993). Hopfield network is recurrent and fully interconnected with each neuron feeding its output in all others. The concept was that all the neurons would transmit signals back and forth to each other in a closed feedback loop until their state become stable. The Hopfield topology is illustrated in Figure 3.7.

Figure 3.8 provides examples of a Hopfiled network with nine neurons. The operation is as follows:

Assume that there are three 9-dimensional vectors α, β and γ.

$$\alpha = (101010101) \ \beta = (110011001) \ \gamma = (010010010). \tag{3.2.12}$$

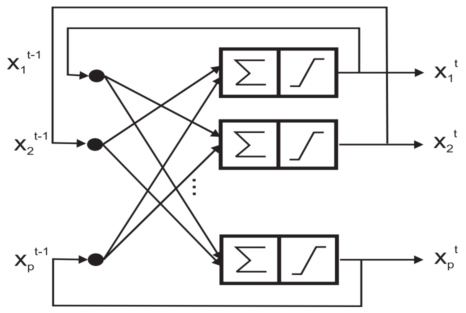


Figure 3.7: Hopfield network architecture

The substitution of -1 for each 0 in these vectors transforms then into bipolar form $\alpha^*, \beta^*, \gamma^*$.

$$\begin{aligned} \alpha^* &= (1 - 11 - 11 - 11 - 11)' \\ \beta^* &= (11 - 1 - 111 - 1 - 11)' \\ \gamma^* &= (-11 - 1 - 11 - 1 - 11 - 1)'. \end{aligned} \tag{3.2.13}$$

The weight matrix is

$$\boldsymbol{W} = \alpha^* \alpha^{*\prime} + \beta^* \beta^{*\prime} + \gamma^* \gamma^{*\prime} \tag{3.2.14}$$

when the main diagonal is zeroed (no neuron feeding into itself). This becomes the matrix in Figure 3.8 of the example.

	A	B	C	D	E	F	G	H	I
From A	0	-1	1	-1	1	1	1	-3	3
B	-1	0	-3	-1	1	1	-3	1	-1
C	1	-3	0	1	-1	-1	3	-1	1
D	-1	-1	1	0	-3	1	1	1	-1
E	1	1	-1	-3	0	-1	-1	-1	1
F	1	1	-1	1	-1	0	-1	-1	1
G	1	-3	3	1	-1	-1	0	-1	1
H	-3	1	-1	1	-1	-1	-1	0	-3
I	3	-1	1	-1	1	1	1	-3	0

Synaptic weight matrix, nodes $A - I$

Start	a	b	Start	a	Start	a	b
$A = 1$	$A = 1$	$A = 1$	$A = 1$	$A = 1$	$A = 1$	$A = 1$	$A = 1$
$B = 1$	$B \rightarrow 0$	$B = 0$	$B = 1$	$B = 1$	$B = 0$	$B = 0$	$B = 0$
$C = 1$	$C = 1$	$C = 1$	$C = 0$	$C = 0$	$C = 0$	$C \rightarrow 1$	$C = 1$
$D = 1$	$D \rightarrow 0$	$D \rightarrow 1$	$D = 0$	$D = 0$	$D = 1$	$D = 1$	$D = 1$
$E = 1$	$E \rightarrow 0$	$E = 0$	$E = 1$	$E = 1$	$E = 0$	$E = 0$	$E = 0$
$F = 1$	$F \rightarrow 0$	$F \rightarrow 1$	$F = 1$	$F = 1$	$F = 0$	$F \rightarrow 1$	$F = 1$
$G = 1$	$G = 1$	$G = 1$	$G = 0$	$G = 00$	$G = 1$	$G = 1$	$G = 1$
$H = 1$	$H \rightarrow 0$	$H = 0$	$H = 0$	$H = 0$	$H = 0$	$H = 0$	$H = 0$
$I = 1$	$I = 1$	$I = 1$	$I = 0$	$I \rightarrow 1$	$I = 0$	$I \rightarrow 1$	$I = 1$

Example 1 Example 2 Example 3

Figure 3.8: Example of Hopfield network (The matrix shows the synaptic weights between neurons. Each example shows an initial state for the neurons (representing an input vector). State transitions are shown by arrows, as individual nodes are updated one at a time in response to the changing states of the other nodes. In example 2, the network output converges to a stored vector. In examples 1 and 3, the stable output is a spontaneous attractor, identical to none of the three stored vectors. The density of 1-bits in the stored vectors is relatively high and the vectors share enough is in common bit locations to put them far from orthogonality - all of which may make the matrix somewhat quirky, perhaps contributing to its convergence toward the spontaneous attractor in examples 1 and 3.)

$$\begin{array}{ccc} \cdot & \# & \cdot \\ \# & \cdot & \# \\ \# & \# & \# \\ \# & \cdot & \# \\ \# & \cdot & \# \end{array} \qquad \begin{array}{ccc} \cdot & \# & \# \\ \# & \cdot & \cdot \\ \# & \cdot & \cdot \\ \# & \cdot & \cdot \\ \cdot & \# & \# \end{array}$$

$$(-1,1) \qquad (1,1)$$

The study of the dynamics of the Hopfield network and its continuous extension is an important subject which includes the study of network capacity (How many neurons are need to store a certain number of pattern? Also it shows that the bipolar form allows greater storage) energy (or Lyapunov) function of the system. These topics are more related to applications to Optimization than to Statistics.

We end this section with an example from Fausett (1994) on comparison of data using a BAM (Bidirectional Associative Memory) network. The BAM network is used to associate letters with simple bipolar codes.

Consider the possibility of using a (discrete) BAM network (with bipolar vectors) to map two simple letters (given by 5×3 patterns) to the following bipolar codes:

The weight matrices are:

$$\begin{array}{ccc} \text{(to store } A \to -11) & (C \to 11) & (\boldsymbol{W}\text{, to store both)} \\ \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} & \begin{bmatrix} -1 & -1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ -1 & -1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} & \begin{bmatrix} 0 & -2 \\ 0 & 2 \\ 2 & 0 \\ 0 & 2 \\ 0 & -2 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ -2 & 0 \\ 0 & 2 \\ 0 & -2 \\ -2 & 0 \\ -2 & 0 \\ 2 & 0 \\ 0 & 2 \end{bmatrix} \end{array} \tag{3.2.15}$$

To illustrate the use of a BAM, we first demonstrate that the net gives the correct Y vector when presented with the $\boldsymbol{x}$ vector for either pattern A or the pattern

C:

INPUT PATTERN A

$$\begin{bmatrix}-11 & -11 & -111111 & -111 & -11\end{bmatrix}\boldsymbol{W} = (-14, 16) \to (-1, 1) \tag{3.2.16}$$

INPUT PATTERN C

$$\begin{bmatrix}-1111 & -1 & -11 & -1 & -11 & -1 & -1 & -11\end{bmatrix}\boldsymbol{W} = (14, 16) \to (1, 1) \tag{3.2.17}$$

To see the bidirectional nature of the net, observe that the Y vectors can also be used as input. For signals sent from the Y-layer to the X-layer, the weight matrix is the transpose of the matrix $\boldsymbol{W}$, i.e.,

$$\boldsymbol{W}^T = \begin{bmatrix}0 & 0 & 2 & 0 & 0 & -2 & 0 & -2 & -2 & 0 & 0 & -2 & -2 & 2 & 0\\ 0 & 0 & 2 & 0 & 0 & -2 & 0 & -2 & -2 & 0 & 0 & -2 & -2 & 2 & 0\end{bmatrix} \tag{3.2.18}$$

For the input vector associated with pattern A, namely, (-1,1), we have

$$\begin{aligned}
&(-1, 1)\boldsymbol{W}^T = \\
&(-1, 1)\begin{bmatrix}0 & 0 & 2 & 0 & 0 & -2 & 0 & -2 & -2 & 0 & 0 & -2 & -2 & 2 & 0\\ 0 & 0 & 2 & 0 & 0 & -2 & 0 & -2 & -2 & 0 & 0 & -2 & -2 & 2 & 0\end{bmatrix} \\
&= \begin{bmatrix}-2 & 2 & -2 & 2 & -2 & 2 & 2 & 2 & 2 & 2 & -2 & 2 & 2 & -2 & 2\end{bmatrix} \\
&\to \begin{bmatrix}-1 & 1 & -1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & -1 & 1\end{bmatrix}
\end{aligned} \tag{3.2.19}$$

This is pattern A.

Similarly, if we input the vector associated with pattern C, namely, (1,1), we obtain

$$\begin{aligned}
&(-1, 1)\boldsymbol{W}^T = \\
&(-1, 1)\begin{bmatrix}0 & 0 & 2 & 0 & 0 & -2 & 0 & -2 & -2 & 0 & 0 & -2 & -2 & 2 & 0\\ -2 & 2 & 0 & 2 & -2 & 0 & & 0 & 0 & 2 & -2 & 0 & 0 & 0 & 2\end{bmatrix} \\
&= \begin{bmatrix}-2 & 2 & 2 & 2 & -2 & -2 & 2 & -2 & -2 & 2 & -2 & -2 & -2 & -2 & 2\end{bmatrix} \\
&\to \begin{bmatrix}-1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & 1 & 1\end{bmatrix}
\end{aligned} \tag{3.2.20}$$

which is pattern C.

The net can also be used with noisy input for the x vector, the y vector, or both.

3.3 Radial Basis Function Networks

The radial basis function is a single hidden layer feed forward network with linear output transfer function and nonlinear transfer function $h(\cdot)$ in the hidden layer. Many types of nonlinearity may be used.

The most general formula for the radial function is

$$h(x) = \Phi\left((x-c)R^{-1}(x-c)\right) \tag{3.3.1}$$

where Φ is the function used (Gaussian, multiquadratic, etc.), c is the center and R the metric. The common functions are: Gaussian $\Phi(z) = e^{-z}$, Multiquadratic $\Phi(z) = (1+z)^{1/2}$, Inverse multiquadratic $\Phi(z) = (1+z)^{-1/2}$ and the Cauchy $\Phi(z) = (1+z)^{-1}$.

If the metric is Euclidean $R = r^2 I$ for some scalar radius and the equation simplifies to

$$h(x) = \Phi\left(\frac{(x-c)^T(x-c)}{r^2}\right). \tag{3.3.2}$$

The essence of the difference between the operation of radial basis function and multilayer perceptron is shown in Figure 3.9 for a hypothetical classification of psychiatric patients.

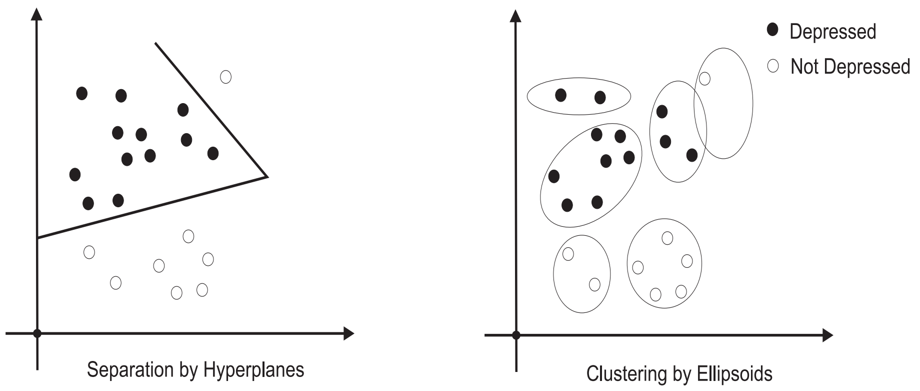


Figure 3.9: Classification by Alternative Networks: multilayer perceptron (left) and radial basis functions networks (right)

Multilayer perceptron separates the data by hyperplanes while radial basis function cluster the data into a finite number of ellipsoids regions.

A simplification is the 1-dimensional input space in which case we have

$$h(x) = \Phi\left(\frac{(x-c)^2}{\sigma^2}\right). \tag{3.3.3}$$

Figure 3.10 illustrate the alternative forms of $h(x)$ for $c = 0$ and $r = 1$.

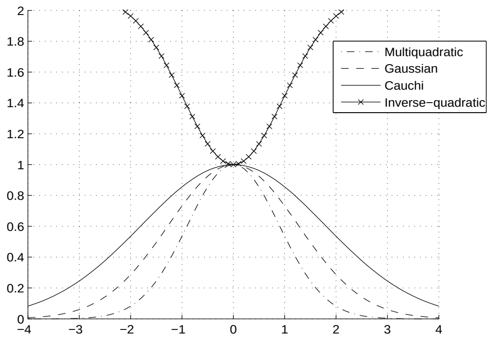


Figure 3.10: Radial basis functions h(.)

To see how the RBFN works we consider some examples (Wu and McLarty, 2000). Consider two measures (from questionnaires) of mood to diagnose depression. The algorithm that implemented the radial basis function application determine that seven cluster sites were needed for this problem. A constant variability term, $\sigma^2 = 1$, was used for each hidden unit. The Gaussian function was used for activation of the hidden layers. Figure 3.11 shows the network.

Suppose we have the scores $x' = (4.8; 1.4)$. The Euclidean distance from this vector and the cluster mean of the first hidden unit $\mu_1' = (0.36; 0.52)$ is

$$D = \sqrt{(4.8 - .36)^2 + (.14 - .52)^2} = .398. \tag{3.3.4}$$

The output of the first hidden layer is

$$h_1(D^2) = e^{-.398^2/2\times(.1)} = .453. \tag{3.3.5}$$

This calculation is repeated for each of the remaining hidden unit. The final output is

$$(.453)(2.64) + (.670)(-.64) + (543)(5.25) + \ldots + (-1)(0.0092) = 0.0544 \tag{3.3.6}$$

Since the output is close to 0, the patient is classified as not depressed.

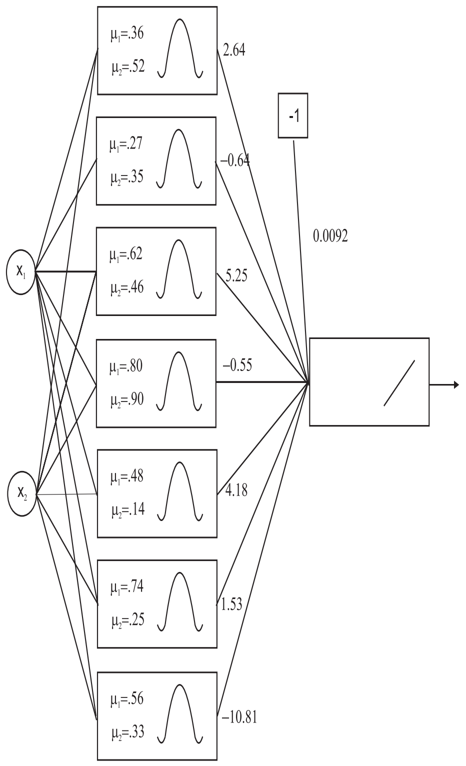


Figure 3.11: Radial basis network

The general form of the output is $y = f(x) = \sum_{i=1}^{m} w_i h_i(x)$, that is the function f is expressed as a linear combination of m fixed functions (called basis function in analogy to the concept of a vector being composed as combination of basis vectors).

Training radial basis function networks proceeds in two steps. First the hidden layer parameters are determined as a function of the input data and then the weights in between the hidden and output layer are determined from the output of the hidden layer and the target data.

Therefore we have a combination of unsupervised training to find the cluster and the cluster parameters and in the second step a supervised training which is linear in the parameters. Usually in the first step the k-means algorithm of clustering is used, but other algorithms can also be used including SOM and ART (chapter 4).

3.4 Wavelet Neural Networks

The approximation of a function in terms of a set of orthogonal basis function is familiar in many areas, in particular in statistics. Some examples are: the Fourier expansion, where the basis consists of sines and cosines of differing frequencies. Another is the Walsh expansion of categorical time sequence or square wave, where the concept of frequency is replaced by that of sequence, which gives the number of "zero crossings" of the unit interval. In recent years there has been a considerable interest in the development and use of an alternative type of basis function: *wavelets*.

The main advantage of wavelets over Fourier expansion is that wavelets is a localized approximator in comparison to the global characteristic of the Fourier expansion. Therefore wavelets have significant advantages when the data is non-stationary.

In this section, we will outline why and how neural network has been used to implement wavelet applications. Some wavelets results are first presented.

3.4.1 Wavelets

To present the wavelet basis of functions we start with a *father wavelet* or *scaling* function ϕ such that

$$\phi(x) = \sqrt{2} \sum_{k} l_k \phi(2x - k) \qquad (3.4.1)$$

usually, normalized as $\int_{-\infty}^{\infty} \phi(x)dx = 1$. A mother wavelet is obtained through

$$\psi(x) = \sqrt{2} \sum h_k \phi(2x - k) \tag{3.4.2}$$

where l_k and h_k are related through

$$h_k = (-1)^k l_k. \tag{3.4.3}$$

The equations 3.4.1 and 3.4.2 are called *dilatation equations*, the coefficients l_k, h_k are low-pass and high-pass filters, respectively.

We assume that these functions generate an orthonormal system of $L_2(\mathcal{R})$, denoted $\{\phi_{j_0,k}(x)\} \cup \{\psi_{jk}(x)\}_{j \geq j_0,k}$ with $\phi_{j_0,k}(x) = 2^{j_0/2}\phi(2^{j_0}x - k),\ \psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k)$ for $j \geq j_0$ the coarse scale.

For any $f \in L_2(\mathcal{R})$ we may consider the expansion

$$f(x) = \sum_{k=-\infty}^{\infty} \alpha_k \phi_{j_0,k}(x) + \sum_{j \geq j_0} \sum_{k=-\infty}^{\infty} \beta_{j,k}\psi_{j,k}(x) \tag{3.4.4}$$

for some coarse scale j_0 and where the true wavelet coefficients are given by

$$\alpha_k = \int_{-\infty}^{\infty} f(x)\phi_{j_0,k}(x)dx \ \beta_{j,k} = \int_{-\infty}^{\infty} f(x)\psi_{j,k}(x)dx. \tag{3.4.5}$$

An estimate will take the form

$$\hat{f}(x) = \sum_{k=-\infty}^{\infty} \hat{\alpha}_k \phi_{j_0,k}(x) + \sum_{j \geq j_0} \sum_{k=-\infty}^{\infty} \hat{\beta}_{j,k}\psi(x) \tag{3.4.6}$$

where $\hat{\alpha}_k$ and $\hat{\beta}_k$ are estimates of α_k and β_{jk}, respectively.

Several issues are of interests in the process of obtaining the estimate $\hat{f}(x)$:

(i) the choice of the wavelet basis;

(ii) the choice of thresholding policy (which $\hat{\alpha}_j$ and $\hat{\beta}_{jk}$ should enter in the expression of $\hat{f}(x)$; the value of j_0; for finite sample the number of parameters in the approximation);

(iii) the choice of further parameters appearing in the thresholding scheme;

(iv) the estimation of the scale parameter (noise level).

These issues are discussed in Morettin (1997) and Abramovich et al (2000).

Concerning the choice of wavelet basis, some possibilities includes the Haar wavelet which is useful for categorical-type data. It is based on

$$\begin{aligned} \phi(x) &= 1 \quad 0 \leq x < 1 \\ \psi(x) &= \begin{cases} 1 & 0 \leq x < 1/2 \\ -1 & 1/2 \leq x < 1 \end{cases} \end{aligned} \tag{3.4.7}$$

and the expansion is then

$$f(x) = \alpha_0 + \sum_{j=0}^{J}, \sum_{k=0}^{2^J-1} \beta_{jk}\psi_{jk}(x). \tag{3.4.8}$$

Other common wavelets are: the Shannon with mother function

$$\psi(x) = \frac{\sin(\pi x/2)}{\pi x/2}\cos(3\pi x/2) \tag{3.4.9}$$

the Mortet wavelet with mother function

$$\psi(x) = e^{iwx}e^{-x^2/2} \tag{3.4.10}$$

where $i = \sqrt{-1}$ and w is a fixed frequency. Figure 3.12 shows the plots of this wavelets. Figure 3.13 illustrates the localized characteristic of wavelets.

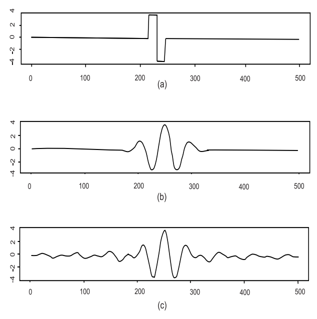


Figure 3.12: Some wavelets: (a) Haar (b) Morlet (c) Shannon

Figure 3.13 gives an example of a wavelet transform.

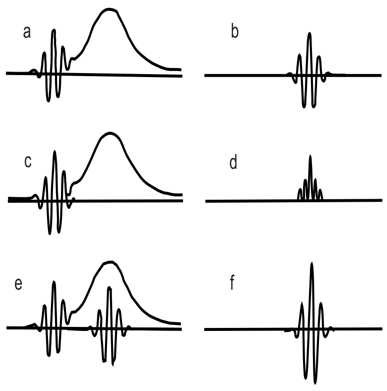


Figure 3.13: Wavelet operation

In figure 3.13 a wavelet (b) is compared successively to different sections of a function (a). The product of the section and the wavelet is a new function; the area delimited by that function is the wavelet coefficient. In (c) the wavelet is compared to a section of the function that looks like the wavelet. The product of the two is always positive, giving the big coefficient shown in (d). (The product of two negative functions is positive.) In (e) the wavelet is compared to a slowly changing section of the function, giving the small coefficients shown in (f). The signal is analyzed at different scales, using wavelets of different widths. "You play with the width of the wavelet in order to catch the rhythm of the signal," says Yves Meyer.

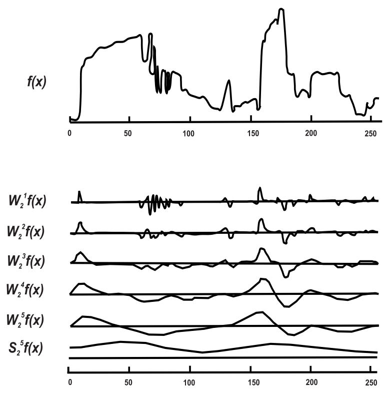


Figure 3.14: An Example of a Wavelet Transform.

Figure 3.4.1 shows: (a) The original signal. (b) The wavelet transform of the signal, over 5 scales, differing by a factor of 2. The finest resolution, giving the smallest details is at the top. At the bottom is the graph of the remaining low frequencies.

3.4.2 Wavelet Networks and Radial Basis Wavelet Networks.

We have seen (Section 2.7 - Kolmogorov Theorem) that neural networks have been established as a general approximation tool for fitting non-linear models from input/output data. On the other hand, wavelets decomposition emerges as powerful tool for approximation. Because of the similarity between the discrete wavelet transform and a one-hidden-layer neural network, the idea of combining both wavelets and neural networks has been proposed. This has resulted in the wavelet network which is discussed in Iyengar et al (2002) and we will summarize here.

There are two main approaches to form wavelet network. In the first, the wavelet component is decoupled from the estimation components of the perceptron. In essence the data is decomposed on some wavelets and this is fed into the neural network. We call this *Radial Basis Wavelet Neural Networks* which is shown in Figures 3.15(a),3.15(b).

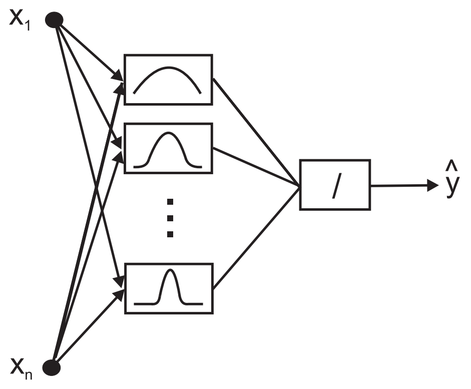
In the second approach, the wavelet theory and neural networks are combined into a single method. In wavelet networks, both the position and dilatation of the wavelets as well as the weights are optimized. The neuron of a wavelet network is a multidimensional wavelet in which the dilatation and translations coefficients are considered as neuron parameters. The output of a wavelet network is a linear combination of several multidimensional wavelets.

Figures 3.16 and 3.17 give a general representation of a wavelet neuron (wavelon) and a wavelet network

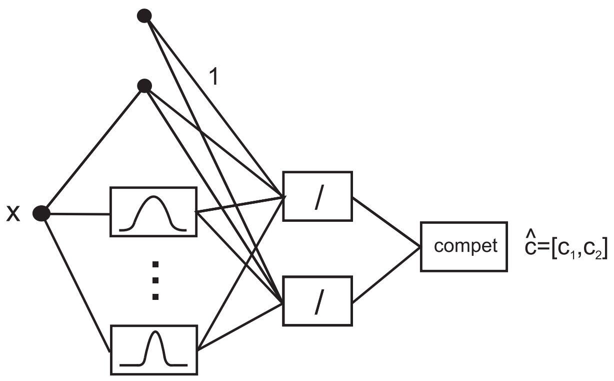
Observation. It is useful to rewright the wavelet basis as:

$$\Psi_{a,b} = |a|^{-1} \left(\frac{x-b}{a} \right) \quad a > 0, -\infty < b < \infty \text{ with } a = 2^j, b = k2^{-j}$$

A detailed discussion of this networks is given in Zhang and Benvensite (1992). See also Iyengar et al (1992) and Martin and Morris (1999) for further references.



(a) Function approximation



(b) Classification

Figure 3.15: Radial Basis Wavelet Network

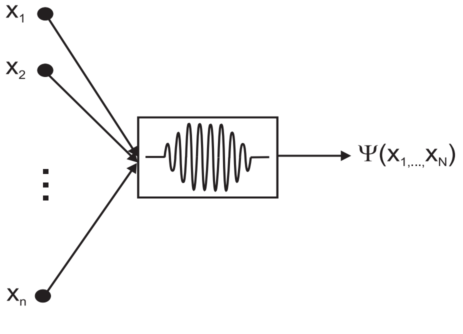


Figure 3.16: Wavelon

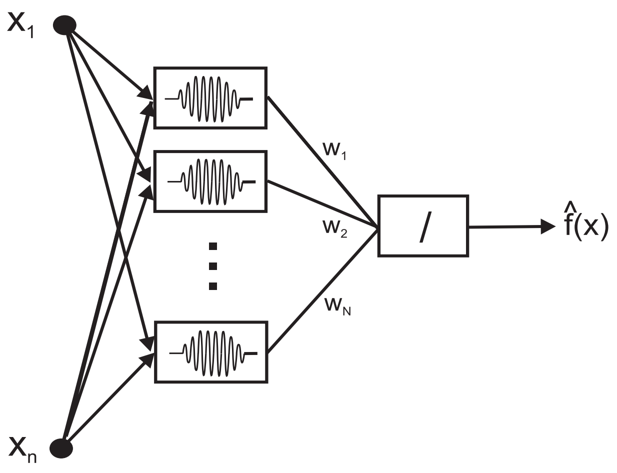


Figure 3.17: Wavelet network - $f(x) = \sum_{i=1}^{N} w_i \Psi\left(\frac{x_i - a_i}{b_i}\right)$

3.5 Mixture of Experts Networks

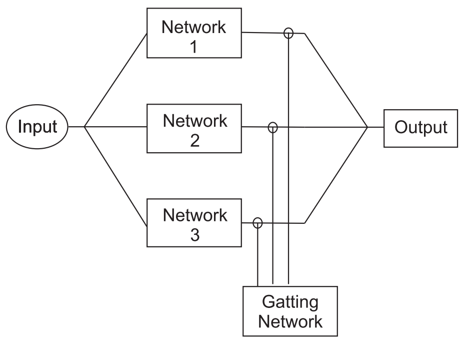
Consider the problem of learning a function in which the form of the function varies in different regions of the input space (see also Radial Basis and Wavelets Networks Sections). These types of problems can be made easier using Mixture of Experts or Modular Networks.

In these type of network we assigned different expert networks to table each of the different regions, and then used an extra "gating" network, which also sees the input data, to deduce which of the experts should be used to determine the output. A more general approach would allow the discover of a suitable decomposition as part of the learning process.

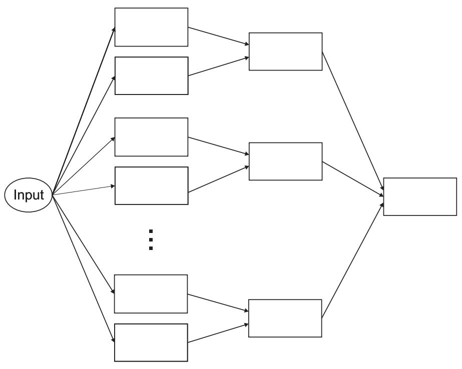
Figure 3.18 shows several possible decompositions

These types of decompositions are related to mixture of models, generalized additive and tree models in statistics (see Section 5.2)

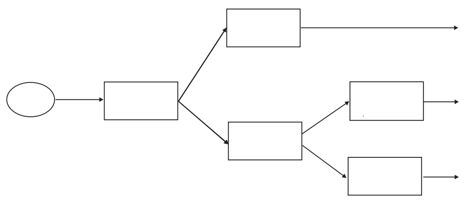
Further details and references can be seen in Mehnotra et al (2000) and Bishop (1995). Applications can be seen in Jordan and Jacobs (1994), Ohno-Machodo (1995), Peng et al (1996), Desai et al (1997), Cheng (1997) and Ciampi and Lechevalier (1997).



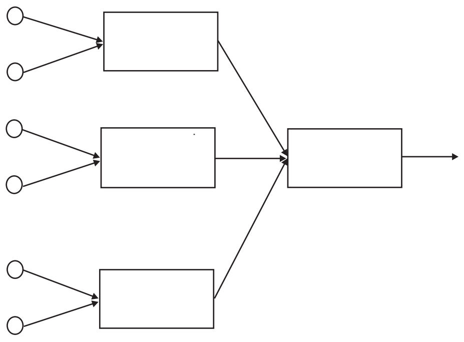
(a) Mixture with gating



(b) Hierarchical



(c) Sucessive refinement



(d) Input modularity

Figure 3.18: Mixture of experts networks

3.6 Neural Network and Statistical Models Interface

The remaining chapters of this book apply these networks to different problems of data analysis in place of statistical models. It is therefore convenient to review the statistical bibliography relating neural network and statistical models and statistical problems in general.

The relationship of Multilayer Feedforward Network with several linear models is presented by Arminger and Enache (1996). They shown that these networks can perform are equivalent to perform: linear regression,linear discrimination, multivariate regression, probit regression, logit regression, generalized linear model, generalized additive models with known smoothing functions, projection pursuit, LISREL (with mixed dependent variables). Other related paper are: Cheng and Titterington (1994) Sarle (1994), Stern (1996), Warner and Misha (1996), Schumacher et al (1996), Vach et al. (1996), De Veux et al (1998).

Attempts to compare performance of neural networks were made extensively but no conclusive results seems to be possible; some examples are Balakrishnan et al (1994), Mangiameli et al (1996) Mingoti and Lima (2006) for cluster analysis and Tu (1996), Ennis et al (1998) and Schwarzer et al (2000) for logistic regression.

Statistical intervals for neural networks were studied in Tibshirani (1996), Huang and Ding (1997), Chryssolouriz et al (1996) and De Veux et al (1998), For Bayesian analysis for neural networks see MacKay (1992), Neal (1996), Lee (1998, 2004), and Rios-Insua and Muller (1996), Paige and Butle (2001) and the review of Titterington (2004).

Theoretical statistical results are presented in Murtagh (1994), Poli and Jones (1994), Jordan (1995), Faraggi and Simon (1995), Ripley (1996), Lowe (1999), Martin and Morris (1999), the review papers Cheng and Titterington (1994), Titterington (2004). Asymptotics and further results can be seen in a series of papers by White (1989a,b,c, 1996), Swanson and White (1995,1997), Kuan and White (1994). Books relating neural networks and statistics are: Smith (1993), Bishop (1996), Ripley (1996), Neal (1996) and Lee (2004).

Pitfalls on the use of neural networks versus statistical modeling is discussed in Tu (1996), Schwarzer et al (2000) and Zhang (2007). Further references on the interfaces: statistical tests, methods (eg survival, multivariate etc) and others are given in the correspondingly chapters of the book.

Chapter 4

Multivariate Statistics Neural Network Models

4.1 Cluster and Scaling Networks

4.1.1 Competitive networks

In this section, we present some network based on competitive learning used for clustering and in a network with a similar role to that of multidimensional scaling (Kohonen map network).

The most extreme form of a competition is the Winner Take All and is suited to cases of competitive unsupervised learning.

Several of the network discussed in this section use the same learning algorithm (weight updating).

Assume that there is a single layer with M neurons and that each neuron has its own set of weights w_j. An input x' is applied to all neurons. The activation function is the identity. Figure 4.1 illustrates the architecture of the Winner Take All network.

The node with the best response to the applied input vector is declared the winner, according to the winner criterion

$$O_l = \min_{j=1 \dots c} \left\{ d^2(y_j, w_j) \right\} \tag{4.1.1}$$

where d is a distance measure of y_j from the cluster prototype or center. The change of weights is calculated according to

$$w_i(k+1) = w_l(k) + \alpha(x^i - w_i(k)) \tag{4.1.2}$$

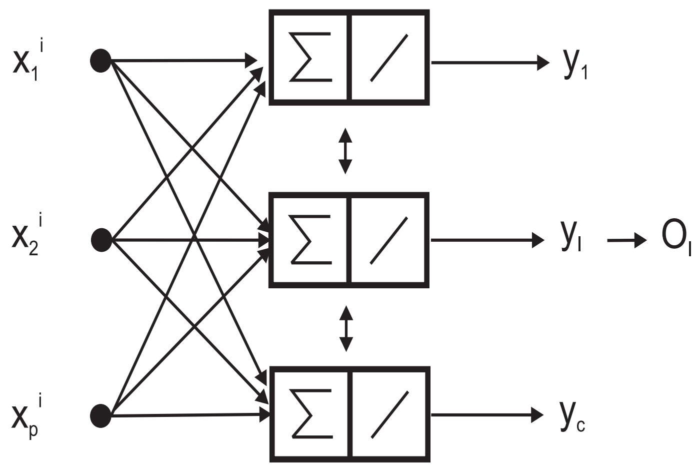


Figure 4.1: General Winner Take All network

where α is the learning rate, and may be decreasing at each iteration.

We now present an example due to Mehrotra et al (2000) to illustrate this simple competitive network.

Let the set n consists of 6 three-dimensional vectors, $x^1, x^2, \ldots, x^6$,

$$\begin{aligned} n = \{&x_1^1 = (1.1, 1.7, 1.8), x_2^2 = (0, 0, 0), x_3^3 = (0, 0.5, 1.5), \\ &x_4^4 = (1, 0, 0), x_5^5 = (0.5, 0.5, 0.5), x_6^6 = (1, 1, 1)\}. \end{aligned} \tag{4.1.3}$$

We begin a network containing three input nodes and three processing units (output nodes), A, B, C. The connection strengths of A, B, C are initially chosen randomly, and are given by the weight matrix

$$W(0) = \begin{pmatrix} w_1 : & 0.2 & 0.7 & 0.3 \\ w_2 : & 0.1 & 0.1 & 0.9 \\ w_3 : & 1 & 1 & 1 \end{pmatrix} \tag{4.1.4}$$

To simplify computations, we use a learning rate $\eta = 0.5$ and update weights equation 4.1.2. We compare squared Euclidean distances to select the winner: $d_{j.l}^2 \equiv d^2(w_j, i_l)$ refers to the squared Euclidean distance between the current position of the processing node j from the lth pattern.

$t = 1$: Sample presented $x_1^1 = (1.1, 1.7, 1.8)$. Squared Euclidean distance between A and x_1^1 : $d_{1.1}^2 = (1.1 - 0.2)^2 + (1.7 - 0.7)^2 + (1.8 - 0.3)^2 = 4.1$. Similarly, $d_{2.1}^2 = 4.4$ and $d_{3.1}^2 = 1.1$.

C is the "winner" since $d^2_{3.1} < d^2_{1.1}$ and $d^2_{3.1} < d^2_{2.1}$. A and B are therefore not perturbed by this sample whereas C moves halfway towards the sample (since $\eta = 0.5$). The resulting weight matrix is

$$W(1) = \begin{pmatrix} w_1: & 0.2 & 0.7 & 0.3 \\ w_2: & 0.1 & 0.1 & 0.9 \\ w_3: & 1.05 & 1.35 & 1.4 \end{pmatrix}. \tag{4.1.5}$$

$t = 2$: Sample presented $x^2_2 = (0, 0, 0), d^2_{1.2} = 0.6, d^2_{2.2} = 0.8, d^2_{3.2} = 4.9$, hence A is the winner. The weights of A is updated. The resulting modified weight vector is $w_1 : (0.1, 0.35, 0.15)$.

Similarly, we have:

$t = 3$: Sample presented $x^3_3 = (0, 0.5, 1.5), d^2_{2.3} = 0.5$ is least, hence B is the winner and is updated. The resulting modified weight vector is $w_2 : (0.05, 0.3, 1.2)$.

$t = 4$: Sample presented $x^4_4 = (1, 0, 0), d^2_{1.4} = 1, d^2_{2.4} = 2.4, d^2_{3.4} = 3.8$, hence A is the winner and is updated: $w_1 : (0.55, 0.2, 0.1)$.

$t = 5$: $x^5_5 = (0.5, 0.5, 0.5)$ is presented, winner A is updated: $w_1(5) = (0.5, 0.35, 0.3)$.

$t = 6$: $x^6_6 = (1, 1, 1)$ is presented, winner C is updated: $w_3(6) = (1, 1.2, 1.2)$.

$t = 7$: x^1_1 is presented, winner C is updated: $w_3(7) = (1.05, 1.45, 1.5)$.

$t = 8$: Sample presented x^2_2. Winner A is updated to $w_1(8) = (0.25, 0.2, 0.15)$.

$t = 9$: Sample presented x^3_3. Winner B is updated to $w_1(9) = (0, 0.4, 1.35)$.

$t = 10$: Sample presented x^4_4. Winner A is updated to $w_1(10) = (0.6, 0.1, 0.1)$.

$t = 11$: Sample presented x^5_5. Winner A is updated to $w_1(11) = (0.55, 0.3, 0.3)$.

$t = 12$: Sample presented x^6_6. Winner C is updated and $w_3(12) = (1, 1.2, 1.25)$.

At this stage the weight matrix is

$$W(12) = \begin{pmatrix} w_1: & 0.55 & 0.3 & 0.3 \\ w_2: & 0 & 0.4 & 1.35 \\ w_3: & 1 & 1.2 & 1.25 \end{pmatrix}. \tag{4.1.6}$$

Node A becomes repeatedly activated by the samples $x^2_2,^4_4$, and x^5_5, node B by x^3_3 alone, and node C by x^1_1 and x^6_6. The centroid of x^2_2, x^4_4, and i_5 is $(0.5, 0.2, 0.2)$, and convergence of the weight vector for node A towards this location is indicated by the progression

$$\begin{aligned} &(0.2, 0.7, 0.3) \to (0.1, 0.35, 0.15) \to (0.55, 0.2, 0.1) \to (0.5, 0.35, 0.3) \to \\ &(0.25, 0.2, 0.15) \to (0.6, 0.1, 0.1) \to (0.55, 0.3, 0.3) \ldots \end{aligned} \tag{4.1.7}$$

To interpret the competitive learning algorithm as a clustering process is attractive, but the merits of doing so are debatable: as illustrated by a simple example were the procedure cluster the number 0, 2, 3, 5, 6 in groups $A = (0, 2), (3, 5), (6)$ instead of the more natural grouping $A = (0), B = (2, 3)$ and $C = (5, 6)$.

For further discussion on the relation of this procedure to "k-means clustering" see Mehrotra et al (2000 pg 168).

4.1.2 Learning Vector Quantization - LVQ

Here again we present the results Mehnotra et al (2000 pg 173).

Unsupervised learning and clustering can be useful preprocessing steps for solving classification problems. A learning vector quantizer (LVQ) is an application of winner-take-all networks for such tasks, and illustrates how an unsupervised learning mechanism can be adapted to solve supervised learning tasks in which class membership is known for every training pattern.

Each node in an LVQ is associated with an arbitrarily chosen class label. The number of nodes chosen for each class is roughly proportional to the number of training patterns that belong to that class, making the assumption that each cluster has roughly the same number of patterns. The new updating rule may be paraphrased as follows.

When pattern i from class $C(i)$ is presented to the network, let the winner node j^* belong to class $C(j^*)$. The winner j^* is moved towards the pattern i if $C(i) = C(j^*)$ and away from i otherwise.

This algorithm is referred to as LVQ1, to distinguish it from more recent variants of the algorithm. In the LVQ1 algorithm, the weight update rule uses a learning rate $\eta(t)$ that is a function of time t, such as $\eta(t) = 1/t$ or $\eta(t) = a[1 - (t/A)]$ where a and A are positive constants and $A > 1$.

The following example illustrates the result of running the LVQ1 algorithm on the input samples of the previous section.

The data are cast into a classification problem by arbitrary associating the first and last samples with Class 1, and the remaining samples with Class 2. Thus the training set is: $T = \{(x_1; 1), (x_2, 0), \ldots, (x_6; 1)\}$ where $x^1 = (1.1, 1.7, 1, 8), x^2 = (0, 0, 0), x^3 = (0, 0.5, 1.5), x^4 = (1, 0, 0), x^5 = (0.5, 0.5, 0.5)$, and $x^6 = (1, 1, 1)$.

The initial weight matrix is

$$W(0) = \begin{pmatrix} w_1 : & 0.2 & 0.7 & 0.3 \\ w_2 : & 0.1 & 0.1 & 1.9 \\ w_3 : & 1 & 1 & 1 \end{pmatrix} \tag{4.1.8}$$

Since there are twice as many samples in Class 2 as in Class 1, we label the first node (w_1) as associated with Class 1, and the other two nodes with Class 2. Let $\eta(t) = 0.5$ until $t = 6$, then $\eta(t) = 0.25$ until $t = 12$, and $\eta(t) = 0.1$ thereafter. Only the change in a single weight vector in each weight update iteration of the network, instead of writing out the entire weight matrix.

1. Sample x_1^1, winner w_3 (distance 1.07), w_3 changed to (0.95, 0.65, 0.60).
2. Sample x_2^2, winner w_1 (distance 0.79), w_1 changed to (0.30, 1.05, 0.45).
3. Sample x_3^3, winner w_2 (distance 0.73), w_2 changed to (0.05, 0.30, 1.20).
4. Sample x_4, winner w_3 (distance 0.89), w_3 changed to (0.97, 0.33, 0.30).

... ...

156. Sample x_6, winner w_1 (distance 0.56), w_1 changed to (1.04, 1.33, 1.38).
157. Sample x_1, winner w_1 (distance 0.57), w_1 changed to (1.05, 1.37, 1.42).
158. Sample x_2, winner w_3 (distance 0.58), w_3 changed to (0.46, 0.17, 0.17).
159. Sample x_3, winner w_2 (distance 0.02), w_2 changed to (0.00, 0.49, 1.48).
160. Sample x_4, winner w_3 (distance 0.58), w_3 changed to (0.52, 0.15, 0.15).
161. Sample x_5, winner w_3 (distance 0.50), w_3 changed to (0.52, 0.18, 0.18).
162. Sample x_6, winner w_1 (distance 0.56), w_1 changed to (1.05, 1.33, 1.38).

Note that associations between input samples and weight vectors stabilize by the second cycle of pattern presentations, although the weight vectors continue to change.

Mehrotra et all also mentioned a variation called the LVQ2 learning algorithm with a different learning rule used instead and an update rule similar to that of LVQ1.

Some application of the procedure are: Pentapoulos et al (1998) apply a LVQ network to discriminate benign from malignant cells on the basis of the extracted morphometric and textural features. Another LVQ network was also applied in an attempt to discriminate at the patient level. The data consisted of 470 samples of voided urine from an equal number of patients with urothelial lesions. For the purpose of the study 45.452 cells were measured. The training sample used 30% of the patients and cells respectively. The study included 50 cases of lithiasis, 61 case of information, 99 cases of benign prostatic hyperplasia, 5 cases of city carcinoma, 71 case of grade I transitional cell carcinoma of the bladder (TCCB) and 184 cases of grade II and grade III TCCB.

The application enable the correct classification of 95.42% of the benign cells and 86.75% of the malignant cells. At the patient level the LVQ network enable the correct classification of 100% of benign cases and 95.6% of the malignant cases. The overall accuracy rate was 90.63% and 97.57%, respectively. Maksimovic and Popovic (1999) compared alternative networks to classify arm movements in tetraplegics. The following hand movements were studied: I - up/down proximal to the body on the lateral side; II - left/right above the level of the shoulder; III - internal/external rotation of the upper arm (humerus).

Figure 4.2 presents the correct classification percentual for the neural networks used.

Vuckovic et al (2002) use neural networks to classify alert vs drowsy states from 1's long sequences of full spectrum EEG in an arbitrary subject. The ex-

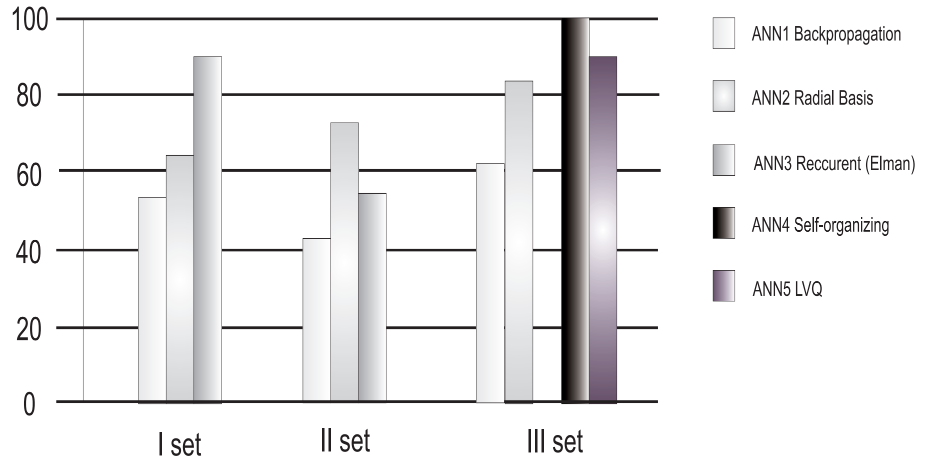


Figure 4.2: Success of classification for all ANNs and all tested movement trials

perimental data was collected on 17 subjects. Two expert in EEG interpretation provided the expertise to train the ANN. The three ANN used in the comparison are: one layer perceptron with identity activation (Widrow-Hoff optimization), perceptron with sigmoid activation (Levenberg-Marquart optimization) and LVQ network.

The LVQ network gives the best classification. For validation 12 volunteers were used and the matching between the human assessment and the network was $94.37 \pm 1.95\%$ using the t statistics.

4.1.3 Adaptive Resonance Theory Networks - ART

Adaptive Resonance Theory (ART) models are neural networks that perform clustering and can allow the number of cluster to vary. The major difference between ART and other clustering methods is that ART allows the user to control the degree of similarity between members of the same cluster by means of a user-defined constant called the *vigilance parameter*.

We outline in this section the simpler version of the ART network called ART1. The architecture of an ART1 network shown in Figure 4.3 consists of two layer of neurons. The F_1 neurons and the F_2 (cluster) neurons together with a reset unit to control the degree of similarity of patterns or sample elements placed on the same cluster unit.

The ART1 networks accepts only binary inputs. For continuous input the ART2 network was developed with a more complex F_1 layer to accommodate continuous input.

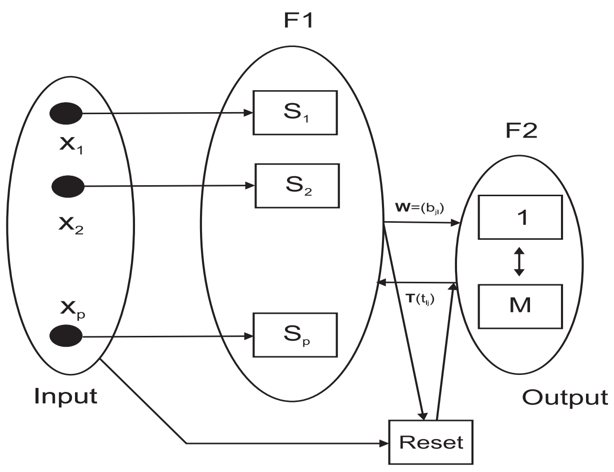


Figure 4.3: Architecture of ART1 network

The input layer, F_1 receives and holds the input patters, the second layer, F_2 responds with a pattern associated with the given input. If this returned pattern is sufficient similar to the input pattern then there is a match. But if the difference is substantial, then the layers communicate until a match is discovered, otherwise a new cluster of patterns is formed around the new input vector.

The training algorithm is shown in Figure 4.4 (Mehrotra et al (2000) or Fauzett (1994)).

The following example of Mehnotra et al (2000) illustrates the computations.

Consider a set of vectors $\{(1,1,0,0,0,0,1), (0,0,1,1,1,1,0), (1,0,1,1,1,1,0),$ $(0,0,0,1,1,1,0)$, and $(1,1,0,1,1,1,0)\}$ to be clustered using the ART1 algorithm. Let the vigilance parameter be $\rho = 0.7$.

We begin with a single node whose top-down weights are all initialized to 1, i.e., $t_{l,1}(0) = 1$, and bottom-up weights are set to $b_{1,l}(0) = \dfrac{1}{8}$. Here $n = 7$ and initially $m = 1$. Given the first input vector, $(1,1,0,0,0,0,1)$, we compute

$$y_1 = \frac{1}{8} \times 1 + \frac{1}{8} \times 1 + \frac{1}{8} \times 0 + \ldots + \frac{1}{8} \times 0 + \frac{1}{8} \times 1 = \frac{3}{8}, \tag{4.1.9}$$

and y_1 is declared the uncontested winner. Since

$$\frac{\sum_{l=1}^{7} t_{l,1} x_l}{\sum_{l=1}^{7} x_l} = \frac{3}{3} = 1 > 0.7, \tag{4.1.10}$$

```
– Initialize each top-down weight $t_{l,j}(0) = 1$;
– Initialize each bottom-up weight $b_{j,l}(0) = \frac{1}{n+1}$;
– while the network has not stabilized, do
    1. Present a randomly chosen pattern $x = (x_1, \ldots, x_n)$ for learning.
    2. Let the active set $A$ contain all nodes; calculate $y_j = b_{j,l}x_1 + \ldots + b_{j,n}x_n$ for each
       node $j \in A$;
        (a) Let $j^*$ be a node in $A$ with largest $y_j$, with ties being broken arbitrarily;
        (b) Compute $s^* = (s_1^*, \ldots, s_n^*)$ where $s_l^* = t_{l,j^*}x_l$;
        (c) Compare similarity between $s^*$ and $x$ with the given vigilance parameter $\rho$:
            if $\frac{\sum_{l=1}^n s_l^*}{\sum_{l=1}^n x_l} \leq \rho$ then remove $j^*$ from set $A$
            else associate $x$ with node $j^*$ and update weights:

                $b_{j^*,l}\,(\text{new}) = \frac{t_{l,j^*}\,(\text{old})\,x_l}{0.5 + \sum_{l=1}^n t_{l,j^*}\,(\text{old})\,x_l}$
                $t_{l,j^*}\,(\text{new}) = t_{l,j^*}\,(\text{old})\,x_l$

            until $A$ is empty or $x$ has been associated with some node $j$;
    3. If $A$ is empty, then create a new node whose weight vector coincides with the current
       input pattern $x$;
  end-while
```

Figure 4.4: Algorithm for updating weights in ART1

the vigilance condition is satisfied and the updated weights are

$$b_{1,l}(1) = \begin{cases} \dfrac{1}{0.5+3} = \dfrac{1}{3.5} & \text{for } l = 1,2,7; \\ 0 & \text{otherwise.} \end{cases} \tag{4.1.11}$$

Likewise,

$$t_{l,1}(1) = t_{l,1}(0)x_l. \tag{4.1.12}$$

These equations yield the following weight matrices.

$$B(1) = \left[\frac{1}{3.5}\ \frac{1}{3.5}\ 0\ 0\ 0\ 0\ \frac{1}{3.5}\right]^T, \qquad T(1) = \left[1\ 1\ 0\ 0\ 0\ 0\ 1\right]^T \tag{4.1.13}$$

Now we present the second sample (0,0,1,1,1,1,0). This generates $y_1 = 0$, but the uncontested winner fails to satisfy the vigilance threshold since $\sum_l t_{l,1}x_l / \sum_l x_l = 0 < 0.7$. A second node must hence be generated, with top-down weights identical to the sample, and bottom-up weights equal to 0 in the positions corresponding to the zeroes in the sample, and remaining new bottom-up weights are equal to $1/(0.5+0+0+1+1+1+1+0)$.

The new weight matrices are

$$B(2) = \begin{bmatrix} \frac{1}{3.5} & \frac{1}{3.5} & 0 & 0 & 0 & 0 & \frac{1}{3.5} \\ 0 & 0 & \frac{1}{4.5} & \frac{1}{4.5} & \frac{1}{4.5} & \frac{1}{4.5} & 0 \end{bmatrix}^T \text{ and } T(2) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}^T \tag{4.1.14}$$

When the third vector $(1,0,1,1,1,1,0)$ is presented to this network,

$$y_1 = \frac{1}{3.5} \text{ and } y_2 = \frac{4}{4.5} \tag{4.1.15}$$

are the node outputs, and the second node is the obvious winner. The vigilance test succeeds, because $\sum_{l=1}^{7} t_{l,2}x_l / \sum_{l=1}^{7} x_l = \frac{4}{5} \geq 0.7$. The second node's weights are hence adapted, with each top-down weight being the product of the old top-down weight and the corresponding element of the sample (1,0,1,1,1,1,0), while each bottom-up weight is obtained on dividing this quantity by $0.5 + \sum_l t_{l,2}x_l = 4.5$. This results in no change to the weight matrices, so that $B(3) = B(2)$ and $T(3) = T(2)$.

When the fourth vector (0,0,0,1,1,1,0) is presented to this network,

$$y_1 = \frac{1}{3.5} \text{ and } y_2 = \frac{3}{4.5} \qquad (4.1.16)$$

are the node outputs, and the second node is the obvious winner. The vigilance test succeeds, because $\sum_{l=1}^{7} t_{l,2}x_l / \sum_{l=1}^{7} x_l = \frac{3}{3} \geq 0.7$. The second node's weights are hence adapted, with each top-down weight being the product of the old top-down weight and the corresponding element of the sample (0,0,0,1,1,1,0), while each bottom-up weight is obtained on dividing this quantity by $0.5 + \sum_l t_{l,2}x_l = 3.5$. The resulting weight matrices are

$$B(4) = \begin{bmatrix} \frac{1}{3.5} & \frac{1}{3.5} & 0 & 0 & 0 & 0 & \frac{1}{3.5} \\ 0 & 0 & 0 & \frac{1}{3.5} & \frac{1}{3.5} & \frac{1}{3.5} & 0 \end{bmatrix}^T, \; T(4) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}^T. \qquad (4.1.17)$$

When the fifth vector (1,1,0,1,1,1,0) is presented to this network,

$$y_1 = \frac{2}{3.5} \text{ and } y_2 = \frac{3}{4.5} \qquad (4.1.18)$$

are the node outputs, and the second node is the obvious winner. The vigilance test fails, because $\sum_{l=1}^{7} t_{l,2}x_l / \sum_{l=1}^{7} x_l = \frac{3}{5} < 0.7$. The active set A is hence reduced to contain only the first node, which is the new winner (uncontested). The vigilance test fails with this node as well, with $\sum_l t_{l,1}x_l / \sum_{l=1}^{7} x_l = \frac{2}{5} \leq 0.7$. A third node is hence created, and the resulting weight matrices are

$$B(5) = \begin{bmatrix} \frac{1}{3.5} & \frac{1}{3.5} & 0 & 0 & 0 & 0 & \frac{1}{3.5} \\ 0 & 0 & 0 & \frac{1}{3.5} & \frac{1}{3.5} & \frac{1}{3.5} & 0 \\ \frac{1}{5.5} & \frac{1}{5.5} & 0 & \frac{1}{5.5} & \frac{1}{5.5} & 0 & \end{bmatrix}^T, \; T(5) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}^T \qquad (4.1.19)$$

We now cycle through all the samples again. This can performed in random order, but we opt for the same sequence as in the earlier cycle. After the third vector is present, again, the weight matrices are modified to the following:

$$B(8) = \begin{bmatrix} \frac{1}{3.5} & \frac{1}{3.5} & 0 & 0 & 0 & 0 & \frac{1}{3.5} \\ 0 & 0 & 0 & \frac{1}{3.5} & \frac{1}{3.5} & \frac{1}{3.5} & 0 \\ \frac{1}{4.5} & 0 & 0 & \frac{1}{4.5} & \frac{1}{4.5} & \frac{1}{4.5} & 0 \end{bmatrix}^T, \; T(8) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}^T . \quad (4.1.20)$$

Subsequent presentation of the samples do not result in further changes to the weights, and $T(8)$ represents the prototypes for the given samples. The network has thus stabilized. For further details see Mehrotra et al (2000).

In that follows we describe some applications of ART neural networks.

Santos (2003) e Santos et al (2006) used neural networks and classification trees in the diagnosis of smear negative pulmonary tuberculosis (SPNT) which account for 30% of the reported cases of Pulmonary Tuberculosis. The data consisted of 136 patients from the health care unit of the Universidade Federal do Rio de Janeiro teaching hospital referred from 3/2001 to 9/2002.

Only symptoms and physical were used for constructing the neural networks and classification. The covariate vector contained 3 continuous variables and 23 binary variables.

In this application an ART neural network identified three groups of patients. In each group the diagnostic was obtained from a one hidden layer feedforward network. The neural networks used had sensibility of 71 to 84% and specificity of 61 to 83% which has slight better than classification tree. Statistical models in literature shows sensibility of 49 to 100% and specificity of 16 to 86% but uses laboratory results. The neural networks of Santos (2003) e Santos et al (2006) used only clinical information.

Rozenthal (1997,see also Rozenthal et al 1998) applied an ART neural network to analyse a set of data from 53 schizophrenic patients (not addicted, physically capable and bellow the age of 50) that answered the DSM-IV (Diagnostic and Statistical Manual for Mental Disorders) and were submitted to neuropsychological tests. There seems to exit at least three functional patterns in schizophrenia. These patterns define such group of symptom or dimension of schizophrenic patients that are: those who have hallucination and disorderly thoughts and self-steem (negative dimension) and those with poor speach and disorderly thoughts (disorganized dimension). This application of neural network (and a classical statistical method of cluster, for comparison) indicate 2 important clusters. One of which, low IQ and negative dimension, keeps itself stable when the number of clusters is increased. It seems to act as an attractor, and also corresponds to the more severe cases and more difficult to respond to treatment. Tables 4.1-4.4 presents the results obtained.

Table 4.1: Presentation of the sample according to level of instruction, IQ and age of onset

Level of Instruction	no. Patients (n=53)	IQ (m ± dp)	age of onset (m ± dp)
College	22	84.91±07.59	21.64±5.58
High school	19	80.84±0.07	19.5 ±6.13
Elementary	12	75.75±08.88	16.67±3.89

Table 4.2: Cluster patterns according to neuropsychological findings

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	RAVLT
Group I (n=20)	77.6	7.7	7.0	-0.05	4.2	11.2	2.7	0.05	4.2	-1.8	-4.2	8.3
Group II (n=3)	84	2.0	24.1	-3.1	4.6	11.6	2.0	100	5.5	-1.8	9.1	11.5

Table 4.3: Three cluster patterns according to neuropsychological findings

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	RAVLT
Group I (n=20)	77.6	7.7	7.0	-0.05	4.2	11.2	2.7	0.05	4.2	-1.8	-4.2	8.3
Group IIa (n=20)	84.7	-0.5	57.1	-14.1	4.5	10.9	1.9	1	5.0	-1.7	-9.1	11.4
Group IIb (n=13)	84.9	3.7	29	-4.0	4.7	12.1	2.1	1	5.8	-1.8	20.8	11.6

Table 4.4: Four cluster patterns according to neuropsychological findings

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	RAVLT
Group I (n=20)	77.6	7.7	7.0	-0.05	4.2	11.2	2.7	0.05	4.2	-1.8	-4.2	8.3
Group IIc (n=14)	80.86	0.81	12.2	6.8	4.7	10.9	2.2	1	6.0	-2.0	0.9	11.4
Group IId (n=11)	85.1	-1.4	51.90	-12.0	4.9	11.6	2.0	1	4.7	-1.7	-21.2	11.2
Group IIe (n=8)	90.0	7.7	43.0	-6.7	4.2	12.5	1.7	1	5.6	-1.6	55.4	12

Chiu and al (2000) developed a self-organizing cluster system for the arterial pulse based on ART neural network. The technique provides at least three novel diagnostic tools in the clinical neurophysiology laboratory. First, the pieces affected by unexpected artificial motions (ie physical disturbances) can be determined easily by the ART2 neural network according to a status distribution plot. Second, a few categories will be created after applying the ART2 network to the input patterns (i.e minimal cardiac cycles). These pulse signal categories can be useful to physicians for diagnosis in conventional clinical uses. Third, the status distribution plot provides an alternative method to assist physicians in evaluating the signs of abnormal and normal automatic control. The method has shown clinical applicability for the examination of the autonomic nervous system.

Ishihara et al (1995) build a Kansey Engineering System based on ART neural network to assist designers in creating products fit for consumers´s underlying needs, with minimal effort. Kansei Engineering is a technology for translating human feeling into product design. Statistical models (linear regression, factor analysis, multidimensional scaling, centroid clustering etc) is used to analyse the feeling-design relationship and building rules for Kansei Engineering Systems. Although reliable, they are time and resource consuming and require expertise in relation to its mathematical constraint. They found that their ART neural network enable quick, automatic rule building in Kansei Engineering systems. The categorization and feature selection of their rule was also slight better than conventional multivariate analysis.

4.1.4 Self Organizing Maps - (SOM) Networks

Self organizing maps, sometimes called topologically ordered maps or Kohonen self-organizing feature maps is a method closely related to multidimensional scaling. The objective is to represent all points in the original space by points in a smaller space, such that distance and proximity relations are preserved as much as possible.

There are m cluster units arranged in a one or two dimensional array.

The weight vector for a cluster neuron serves as an exemplar of the input patterns associated with that cluster. During the self-organizing process, the cluster neuron whose weight vector matches the input pattern most closely (usually, the square of the minimum Euclidean distance) is chosen as the winner. The winning neuron and its neighboring units (in terms of the topology of the cluster neurons update their weights the usual topology in two dimensional are shown in Figure 4.5).

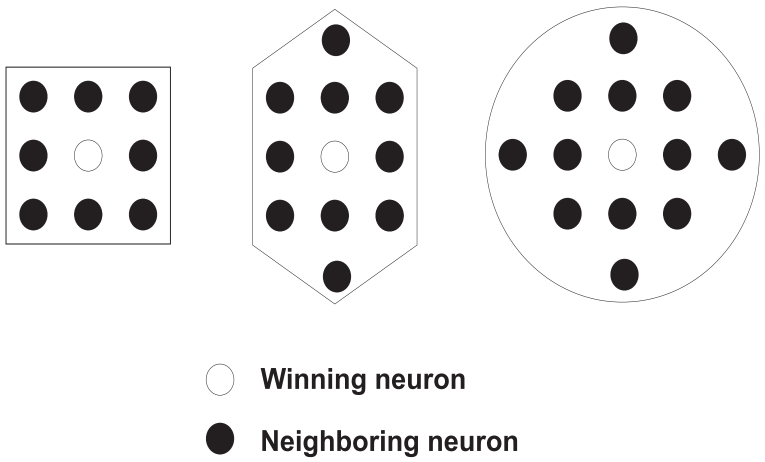


Figure 4.5: Topology of neighboring regions

The architecture of the Kohonen SOM neural network for one and two-dimensional array and its topology is shown in Figures 4.6-4.7 and Figures 4.8-4.9-4.10 for the one dimensional and two dimensional array. Figure 4.11 presents the weight updating algorithm for the Kohonen SOM neural networks (from Fausett 1994) and Figure 4.12, the Mexican Hat interconection for neurons in the algorithm.

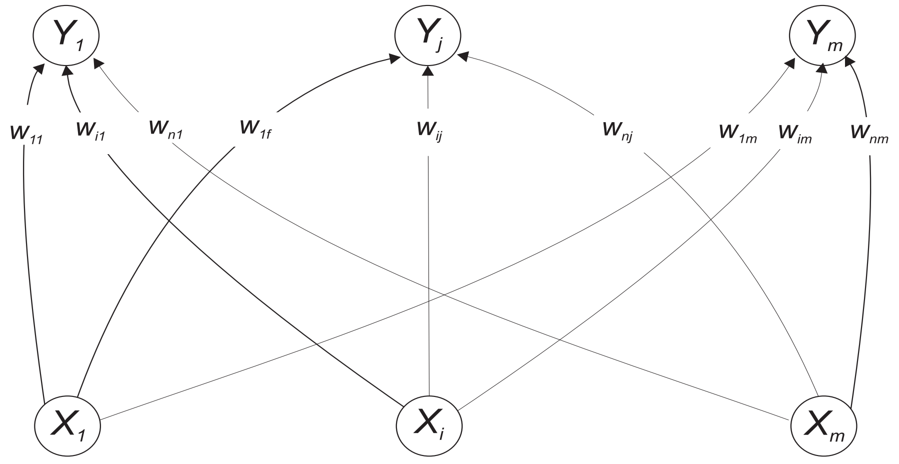


Figure 4.6: Kohonen self-organizing map.

```
*   *   *  {*  (*  [#]  *)  *}  *   *

       { } R=2      ( ) R=1      [ ] R=0
```

Figure 4.7: Linear array of cluster units

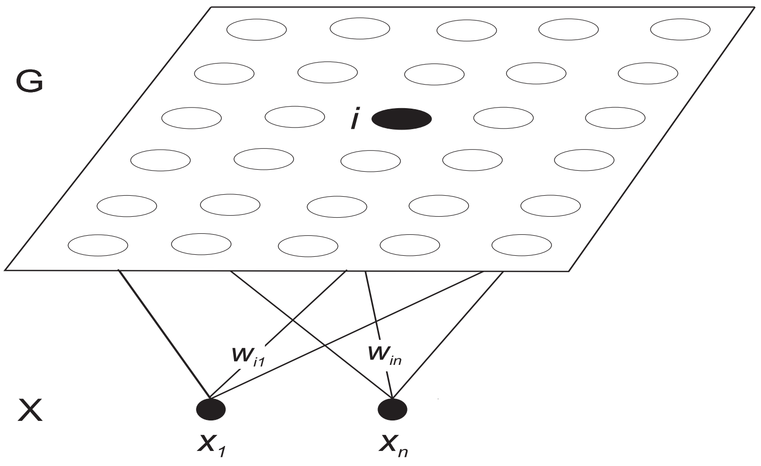


Figure 4.8: The self-organizing map architecture

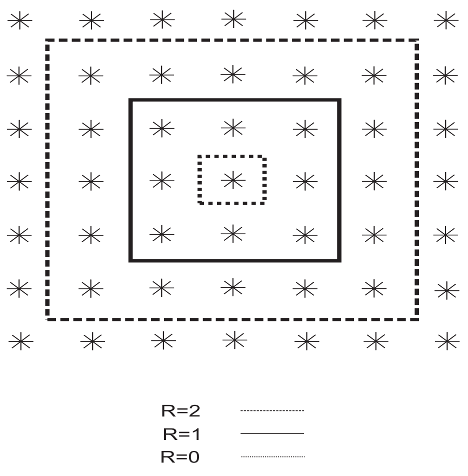


Figure 4.9: Neighborhoods for rectangular grid

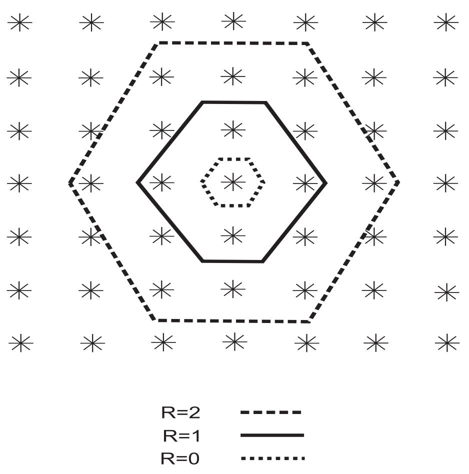


Figure 4.10: Neighborhoods for hexagonal grid

```
Step 0  Initialize weights $w_{ij}$
        Set topological neighborhood parameters.
        Set learning rate parameters.
Step 1  While stopping condition is false, do Steps 2-8.
Step 2      For each input vector $\boldsymbol{x}$, do Steps 3-5.
Step 3          For each $j$, compute:
                $D(j) = \sum_i (w_{ij} - x_i)^2$
Step 4          Find index $J$ such that $D(J)$ is minimum.
Step 5          For all units within a specified neighborhood of $J$ and for all $i$:
                    $w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$
Step 6      Update learning rate.
Step 7  Reduces radius of topological neighborhood at specified times.
Step 8  Test stopping condition.
```

Figure 4.11: Algorithm for updating weights in SOM

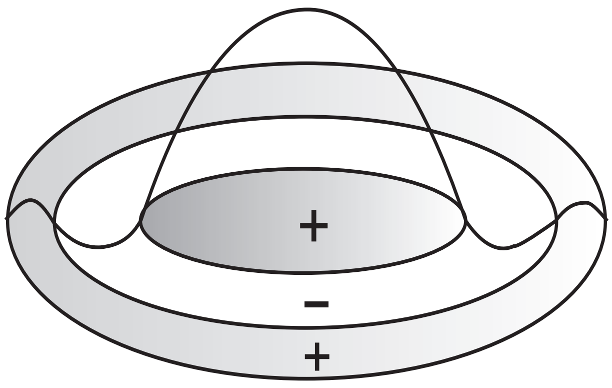


Figure 4.12: Mexican Hat interconnections

Braga et al (2000) sampled 1000 samples with equal probability from two bivariate normal distributions with variances 1 and mean vectors $\mu_1 = (4,4)$,$\mu_2 = (12,12)$. The distribution and the data are shown in Figure 4.13 and 4.14.

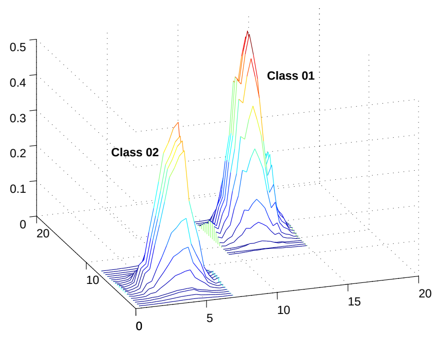


Figure 4.13: Normal density function of two clusters

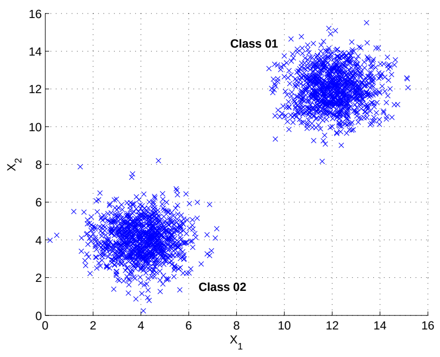


Figure 4.14: Sample from normal cluster for SOM training

The topology was such that each neuron had 4 other neighbouring neurons. The resulting map was expected to preserve the statistical characteristics, ie the visual inspection of the map should indicate how the data is distributed.

Figure 4.15 gives the initial conditions (weights). Figures 4.16- 4.18 presents the map after some interactions.

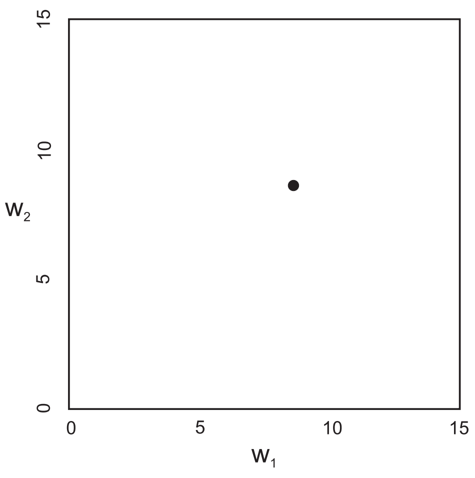


Figure 4.15: Initial weights

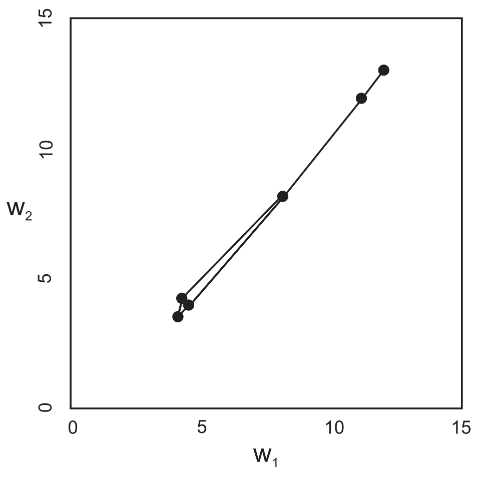


Figure 4.16: SOM weights, 50 iterations

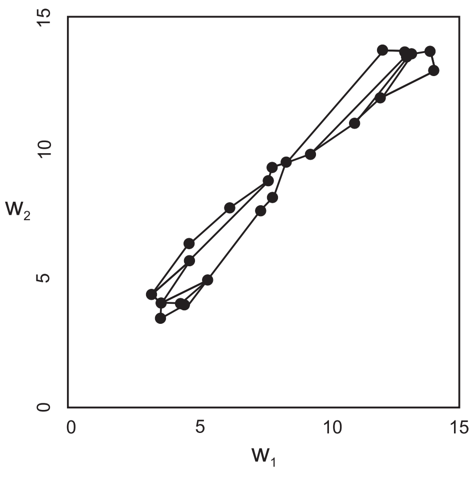


Figure 4.17: SOM weights, 300 iterations

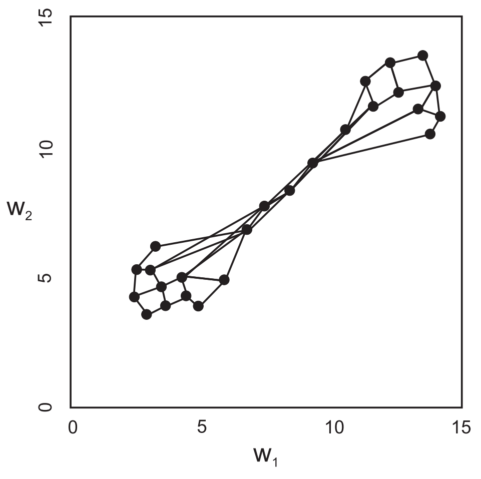


Figure 4.18: SOM weights, 4500 iterations

Lourenço (1998) applied a combination of SOM networks, fuzzy set and classical forecasting methods for the short-term electricity load prediction. The method establish the various load profiles and process climatic variables in a linguistic way, and those from the statistical models. The final model includes a classifier scheme, a predictive scheme and a procedure to improve the estimations. The classifier is implemented via an SOM neural network. The forecaster uses statistical forecasting techniques (moving average, exponential smoothing and ARMA type models). A fuzzy logic procedure uses climatic variables to improve the forecast.

The complete system is shown in Figure 4.19.

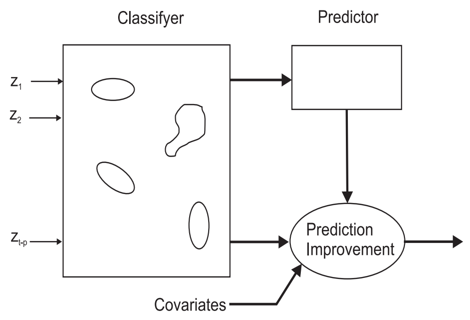


Figure 4.19: Combined forecast system

The classifier used a SOM network with 12 neurons displaced in four rows and three columns. The classification of patterns are shown in Figures 4.20 and Figure 4.21 for the year of 1993.

^{1}Sunday	^{2}Sunday	^{3}Saturday
^{4}Saturday	^{5}Saturday	^{6}Monday to Friday
^{7}Monday to Friday	^{8}Monday to Friday	^{9}Monday to Friday
^{10}Monday to Friday	^{11}Monday to Friday	^{12}Monday to Friday

Figure 4.20: Week days associated with the SOM neurons

The proposed method improved the correct method used in the Brazilian Eletric System, which is important specially in the pick hours.

1	2	3
4	5	6Winter
7Summer	8Spring	9Winter
10Summer	11Autumn	12Winter

Figure 4.21: Seasons associated with the SOM neurons

Draghici and Potter (2003) predict HIV drug resistance using. Since drug resistance is a important factor influencing the failure of current HIV therapies. The ability to predict the drug resistance of HIV protease mutants may be useful in developing more effective and long lasting treatment regimens.

In this work the authors predicted the HIV resistance to two current protease inhibitors. The problem was approached from two perspectives. First, a predictor was constructed based on the structural features of the HIV protease-grug inhibitor complex. A particular structure was represented by its list of contacts between the inhibitor and the protease. Next, a classifier was constructed based on the sequence data of various drug resistent mutants. In both cases SOM networks were first used to extract the important features and cluster the patterns in an unsupervised manner. This was followed by subsequent labelling based on the known patterns in the training set.

The prediction performance of the classifiers was measured by cross-validation. The classifier using the structure information correctly classified previously unseen mutants with and accuracy of between 60 and 70 %. Several architectures were tested on the more abundant sequence of 68 % and a coverage of 69 %. Multiple networks were then combined into various majority voting schemes. The best combination yielded an average of 85% coverage and 78% accuracy on previously unseed data. This is more than two times better than the 33 % accuracy expected from a random classifier.

Hsu et al (2003) describe an unsupervised dynamic hierarchical self-organizing approach, which suggests an appropriate number of clusters, to perform class discovery and marker gene identification in microarray data. In the process of class discovery, the proposed algorithm identifies corresponding sets of predictor genes that best distinguish one class from other classes. The approach integrates merits of hierarchical clustering with robustness against noise know from self-organizing approaches.

The proposed algorithm applied to DNA microarray data sets of two types of cancers has demonstrated its ability to produce the most suitable number of clusters. Further, the corresponding marker genes identified through the unsupervised algorithm also have a strong biological relationship to the specific cancer class. The algorithm tested on leukemia microarray data, which contains three leukemia types, was able to determine three major and one minor cluster. Prediction models built for the four clusters indicate that the prediction strength for the smaller cluster is generally low, therefore labelled as uncertain cluster. Further analysis shows that the uncertain cluster can be subdivided further, and the subdivisions are related to two of their original clusters. Another test performed using colon cancer microarray data has automatically derived two clusters, which is consistent with the number of classes in data (cancerous and normal).

Mangiameh et al (1996) provide a comparison of the performance of SOM network and seven hierarchical cluster methods: single linkage, complete linkage, average linkage, centroid method, Ward´s method, two stage density and k-nearest neighbor. A total of 252 empirical data set was constructed to simulate several levels of imperfections that include dispersion, aberrant observations, irrelevant variables, non-uniform cluster, more detailed description in shown in Tables 4.5-4.7.

Data set	Design factors	# of data sets
Base data	2,3,4 or 5 clusters; 4,6 or 8 variables; low, med or high dispersion	36
Irrelevant variables	Basic data plus 1 or 2 irrelevant variables	72
Cluster density	Basic data plus 10% or 60% density	72
Outliers	Basic data plus 10% or 20% outliers	72

Table 4.5: Data set design

Number of clusters	Average distance (units)
2	5.03
3	5.31
4	5.33
5	5.78

Table 4.6: Average distance between cluster

The experimental results are unambiguous; the SOM network is superior to all seven hierarchical clustering algorithms commonly used today. Furthermore, the

Level of dispersion	Avg. cluster std. dev. (in units)	Range in std.dev.
High	7.72	3
Medium	3.72	6
Low	1.91	12

Table 4.7: Basis for data set construction

performance of the SOM network is shown to be robust across all of these data imperfections. The SOM superiority is maintained across a wide range of ”messy data” conditions that are typical of empirical data sets. Additionally, as the level of dispersion in the data increases, the performance advantage of the SOM network relative to the hierarchical clustering methods increases to a dominant level.

The SOM network is the most accurate method for 191 of the 252 data sets tested, which represents 75.8% of the data. The SOM network ranks first or second in accuracy in 87.3% of the data sets. For the high dispersion data sets, the SOM network is most accurate 90.2% of the time, and is ranked first or second 91.5% of the time. The SOM network frequently has average accuracy levels of 85% or greater, while other techniques average between 35% and 70%. Of the 252 data sets investigated, only six data sets resulted in poor SOM results. These six data sets occurred at low levels of data dispersion, with a dominant cluster containing 60% or more of the observations and four or more total clusters. Despite the relatively poor performance at these data conditions, the SOM network did average 72% of observations correctly classified.

These finds seems to contradict the bad performance of SOM network compared to k-means clustering method applyed in 108 data sets reported by Balakrishman et al (1994).

Carvalho et al (2006) applied the SOM network to gamma ray burst and suggested the existence of 5 clusters of burst. This result was confirmed by a feedfoward network and principal component analysis.

4.2 Dimensional Reduction Networks

In this section we will present some neural networks designed to deal with the problem of dimensional reduction of data.

In several occasions it is useful and even necessary to first reduce the dimension of the data to a manageable size, keeping as much of the original information as possible, and then to proceed with the analysis.

Sometimes, a phenomenon which is in appearance high-dimension is actually governed by few variables (sometimes called ”latent variables” or ”factors”). The redundance, presented in the data collected related to the phenomenon, can be due, for example, to:

– Many of the variables collected may be irrelevant .

– Many of the variables will be correlated (therefore some redundant information is contained in the data), a new set of uncorrelated or independent variables should be found.

An important reason to reduce the dimension of the data is that some authors call: ”the curse of dimensionality and the empty space phenomenon”. The curse of dimensionality phenomenon refers to the fact that in the absence of simplifying assumptions, the sample size needed to make inferences with a given degree of accuracy grows exponentially with the number of variables. The empty space phenomenon responsible for the curse of dimensionality is that high-dimensional spaces are inherently sparse. For example, for a one dimensional standard normal $N(0, 1)$, about 70% of the mass is at points contained in the interval (sphere of radius of one standard deviation around the mean zero. For a 10-dimensional $N(0, I)$, the same (hyper) sphere contain only $0, 02\%$ of the mass, and a radius of more than 3 standard deviations is need to contain 70%.

A related concept in dimensional reduction methods is the intrinsic dimension of a sample, which is defined as the number of independent variables that explain satisfactorily the phenomenon. This is a loosely define concept and a trial and error process us usually used to obtain a satisfactory value for it.

There are several techniques to dimensional reduction and can be classified in two types: with and without exclusion of variables. The main techniques are:

– Selection of variables:

  – Expert opinion.
  – Automatic methods: all possible regressions, best subset regression, backward elimination, stepwise regression, etc.

– Using the original variables:

  – Principal Components.
  – Factor Analysis.
  – Correspondence Analysis.
  – Multidimensional Scaling.

- Nonlinear Principal Components
- Independent Component Analysis
- Others

We are concerned in this book with the latter group of techniques (some of which have already being described: Projection Pursuit, GAM, Multidimensional Scaling, etc).

Now we turn to some general features and common operation on the data matrix to be used in these dimensional reduction methods.

4.2.1 Basic Structure of the Data Matrix

A common operation in most dimensional reduction technique is the decomposition of a matrix of data into its basic structure.

We refer to the data set, the data matrix as $\boldsymbol{X}$, and the specific observation of variable j in subset i, as x_{ij}. The dimension of $\boldsymbol{X}$ is (n x p), corresponding to n observations of p variables ($n > p$).

Any data matrix can be decomposed into its characteristic components:

$$\boldsymbol{X}_{n \text{ x } p} = \boldsymbol{U}_{n \text{ x } p}\boldsymbol{d}_{p \text{ x } p}\boldsymbol{V}'_{p \text{ x } p} \tag{4.2.1}$$

which is call ”single value decomposition”, i.e. SVD.

The $\boldsymbol{U}$ matrix ”summarizes” the information in the rows of $\boldsymbol{X}$, the $\boldsymbol{V}$ matrix ”summarizes” information in the columns of $\boldsymbol{X}$, the $\boldsymbol{d}$ matrix is a diagonal matrix, whose diagonal entries are the singular values and are weigths indicating the relative importance of each dimension in $\boldsymbol{U}$ and $\boldsymbol{V}$ and are ordered from largest to smallest.

If $\boldsymbol{X}$ does not contain redundant information the dimension of $\boldsymbol{U}$, $\boldsymbol{V}$ e $\boldsymbol{d}$ will be equal the minimum dimension p of $\boldsymbol{X}$. The dimension is called rank.

If $\boldsymbol{S}$ is a symmetric matrix the basic structure is:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{d}\boldsymbol{V}' = \boldsymbol{U}\boldsymbol{d}\boldsymbol{U}' = \boldsymbol{V}\boldsymbol{d}\boldsymbol{V}' \tag{4.2.2}$$

Decomposition of $\boldsymbol{X}$, $\boldsymbol{X}.\boldsymbol{X}'$ or $\boldsymbol{X}'.\boldsymbol{X}$ reveals the same structure:

$$\boldsymbol{X}_{nxp} = \boldsymbol{U}\boldsymbol{d}\boldsymbol{V}' \tag{4.2.3}$$

$$\boldsymbol{X}\boldsymbol{X}'_{(nxn)} = \boldsymbol{U}\boldsymbol{d}^2\boldsymbol{U}' \tag{4.2.4}$$

$$\boldsymbol{X}\boldsymbol{X}'_{(pxp)} = \boldsymbol{V}\boldsymbol{d}^2\boldsymbol{V}' \tag{4.2.5}$$

A retangular matrix $\boldsymbol{X}$ can be made into symmetric matrices ($\boldsymbol{X}\boldsymbol{X}'$ and $\boldsymbol{X}'\boldsymbol{X}$) and their eigenstructure can be used to obtain the structure of $\boldsymbol{X}$. If $\boldsymbol{X}\boldsymbol{X}' = \boldsymbol{U}\boldsymbol{D}$ and $\boldsymbol{X}'\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}'$ then $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{V}$.

The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ corresponding to the eigenvalues D_i are also related by: If U_i is an eigenvector of $\boldsymbol{X}\boldsymbol{X}'$ corresponding to D_i then the corresponding eigenvector of $\boldsymbol{X}'\boldsymbol{X}$ corresponding to D_i is equal or proportional to $\boldsymbol{U}_i'\boldsymbol{X}$.

Some dimensional reduction techniques share the same decomposition algorithm SVD. They differ only in the predecomposition transformation of the data and their posdecomposition transformation of the latent variables. The following transformation will be used

– Deviation from the mean:

$$x_{ij} = x_{ij} - \bar{x}_j \tag{4.2.6}$$

where x_j is the column (variable j) mean.

– Standardization

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{S_j} \tag{4.2.7}$$

where $\bar{x}_j$ and S_j are the column (variable j) mean and standard deviation.

– Unit length

$$x^*_{ij} = \frac{x_{ij}}{(\sum x^2_{ij})^{\frac{1}{2}}} \tag{4.2.8}$$

– Double-centering

$$x^*_{ij} = x_{ij} - \bar{x}_i - \bar{x}_j + \bar{x}_{ij} \tag{4.2.9}$$

obtained by subtracting the row and column means and adding back the overall mean.

– For categorical and frequency data

$$x^*_{ij} = \frac{x_{ij}}{(\sum_i x_{ij} \sum_j x_{ij})^{\frac{1}{2}}} = \frac{x_{ij}}{(x_{\bullet j} x_{i\bullet})^{\frac{1}{2}}} \tag{4.2.10}$$

From these transformations the following matrices are obtained:

– Covariance Matrix

$$C = \frac{1}{n}(\boldsymbol{X} - \bar{\boldsymbol{X}})'(\boldsymbol{X} - \bar{\boldsymbol{X}}') \tag{4.2.11}$$

from the mean corrected data.

– R-Correlation Matrix

$$R = \frac{1}{n}\boldsymbol{Z_c'}\boldsymbol{Z_c} \tag{4.2.12}$$

where $\boldsymbol{Z_c}$ indicates the matrix $\boldsymbol{X}$ standardized within columns in the number of observations.

– Q-Correlation Matrix

$$Q = \frac{1}{p}\boldsymbol{Z_r'}\boldsymbol{Z_r} \tag{4.2.13}$$

where $\boldsymbol{Z_r}$ indicates the matrix $\boldsymbol{X}$ standardized within rows, and p is the number of variables.

4.2.2 Mechanics of Some Dimensional Reduction Techniques

Now we outline the relation of the SVD algorithm to the dimensional reduction algorithm, some of which can be computed using neural networks.

4.2.2.1 Principal Components Analysis - PCA

When applying PCA in a set of data the object is:

– to obtain from the original variables a new set of variables (factors, latent variables) which are uncorrelated.

– to hope that a few of this new variables will account for the most of the variation of the data, that is the data can be reasonably represented in lower dimension, and keeping most of the original information

– these new variables can be reasonable interpreted.

The procedure is performed by applying a SVD on the C-Correlation Matrix or in the R-Correlation Matrix. Since PCA is not invariant to transformation usually the R matrix is used.

4.2.2.2 Non-linear Principal Components

Given the data matrix $\boldsymbol{X}$, this procedure consists in performing the SVD algorithm in some function $\phi(\boldsymbol{X})$.

In the statistical literature an early reference is Gnanadesikan (1999) and is closely related to Kernel principal components, Principal Curves and Informax method in Independent Component Analysis. The later will be outlined later.

4.2.2.3 Factor Analysis - FA

Factor analysis has also the aim of reducing the dimensionality of a variable set and it also has the objective of representing a set of variables in terms of a smaller number of hypothetical variables.

The main differences between PCA and FA are:

– PCA decomposes the total variance. In the case of standardized variables, it produces a decomposition of $\boldsymbol{R}$. FA on the other hand finds a decomposition of the reduced matrix $\boldsymbol{R}-\boldsymbol{U}$, where $\boldsymbol{U}$ is a diagonal matrix of the "unique" variances associated with the variables. Unique variances are that part of each variable's variance that has nothing in common with remaining $p-1$ variables.

– PCA is a procedure to decompose the correlation matrix without regard to an underlying model. FA, on the other hand, has an underlying model that rest on a number of assumptions, including normality as the distribution for the variables.

– The emphasis in FA is in explaining X_i as a linear function of hypothetical unobserved common factors plus a factor unique to that variable, while the emphasis in PCA is expressing the principal components as a linear function of the X_i's. Contrary to PCA, the FA model does not provide a unique transformation form variables to factors.

4.2.2.4 Correspondence Analysis - CA

Correspondence analysis represents the rows and columns of a data matrix as points in a space of low dimension, and it is particularly suited to (two-way) contingency tables.

Log-linear model (LLM) is also a method widely used for analysing contingency tables. The choice of method, CA or LLM depends of the type of data to be analysed and on what relations or effects are of most interest. A practical rule for deciding when each method is to be preferred is: CA is very suitable for discovering the inherent structure of contingency tables with variables containing a large number of categories. LLM is particularly suitable for analysing multivariate contingency tables, where the variables containing few categories. LLM analyse mainly the interrelationships between a set of variables, where as correspondence analysis examines the relations between the categories of the variables.

In CA the data consists of an array of frequencies with entries f_{ij}, or as a matrix F.

– The first step is to consider the transformation:

$$h_{ij} = \frac{f_{ij}}{\sqrt{f_{i\bullet} f_{\bullet j}}} \tag{4.2.14}$$

where h_{ij} is the entry for a given cell, f_{ij} the original cell frequency, $f_{i\bullet}$ the total for row i and $f_{\bullet j}$ the total for column j.

– The second step is to find the basic structure of the normalized matrix $\boldsymbol{H}$ with elements (h_{ij}) using SVD. This procedure summary row and column vectors, $\boldsymbol{U}$ and $\boldsymbol{V}$ of $\boldsymbol{H}$ and a diagonal matrix of singular values, $\boldsymbol{d}$, of $\boldsymbol{H}$. The first singular value is always one and the successive values constitute singular values or canonical correlations.

– The third and final step is to rescale the row ($\boldsymbol{U}$) and column ($\boldsymbol{V}$) vectors to obtain the canonical scores. The row ($\boldsymbol{X}$) and column ($\boldsymbol{Y}$) canonical scores are obtained as:

$$X_i = \boldsymbol{U}_i \sqrt{\frac{f_{\bullet\bullet}}{f_{i\bullet}}} \tag{4.2.15}$$

$$Y_i = \boldsymbol{V}_i \sqrt{\frac{f_{\bullet\bullet}}{f_{\bullet j}}} \tag{4.2.16}$$

4.2.2.5 Multidimensional Scaling

Here we will describe the classical or metric method of multidimensional scaling which is closely related to the previous dimensional reduction method, since it also os based on SVD algorithm. Alternative mapping and ordinal methods will not be outlined.

In multidimensional scaling we are given the distances d_{rs} between every para of observations. Given the data matrix $\boldsymbol{X}$, there are several ways to measure the distance between pair of observations. Since many of them produce measure of distances which do not satisfy the triangle inequality, the term dissimilarity is used.

For continuous data $\boldsymbol{X}$, the most common choice is the Euclidean distance: $d^2 = \boldsymbol{X}\boldsymbol{X}'$ This depend on the scale in which the variables are measured, One way out is to use Mahalanobis distance with respect to the covariance matrix $\hat{\Sigma}$ of the observations.

For categorical data a commonly used dissimilarity is based on the simple matching coefficient, that is the proportion C_{rs}, of features which are common to the two observations r and s. As this is between zero and one, the dissimilarity is found to be $d_{rs} = 1 - c_{rs}$.

For ordinal data we use the rank as if they were continuous data after rescaling to the range (0,1) so that every original feature is given equal weight.

For mixture of continuous, categorical and ordinal features a widely used procedure is: for each feature define a dissimilarity d^f_{rs} and an indicator I^f_{rs} which is one only if feature f is recorded for both observation. Further $I^f_{rs} = 0$ if we have a categorical feature and an absence- absence. Then

$$d_{rs} = \frac{\sum_f I^f_{rs} d^f_{rs}}{\sum I^f_{rs}} \tag{4.2.17}$$

In the classical or metric method of multidimensional scaling, also known as principal coordinate analysis, we assume that the dissimilarities were derived as Euclidean distances between n points in p dimensions. The steps are as follows:

– From the data matrix obtain the matrix of distances $\boldsymbol{T}$ ($\boldsymbol{X}\boldsymbol{X}'$ or $\boldsymbol{X}\Sigma\boldsymbol{X}'$).

– Double centering the dissimilarity matrix $\boldsymbol{T}$, say $\boldsymbol{T}^*$.

– Decompose the matrix $\boldsymbol{T}^*$ using the SVD algorithm.

– From the p (at most) non-zero vectors represent the observations in the space of two (or three) dimensions.

4.2.3 Independent Component Analysis - ICA

The ICA model will be easier to explain if the mechanics of a cocktail party are first described. At a cocktail party there are partygoers or speaker holding conversations while at the same time there are microphones recording or observing the speakers also called underlying source or component.

At a cocktail party, there are p microphones that record or observe m partygower or speaker at n instants. This notation is consistent with traditional Multivariate Statistics. The observed conversation consists of mixtures of true unobserved conversations. The microphones do not observe the observed conversation in isolation. The recorded conversation are mixed. The problem is to unmix or recover the original conversation from the recorded mixed conversation.

The relation of ICA with other methods of dimension reduction is shown in Figure 4.22 as discussed in Hyvarinen (1999). The lines show close connections, and the text next to the lines show the assumptions needed for the connection.

Before we formalize the ICA method we should mention that the roots of basic ICA can be traced back to the work of Darmois in the 1950 (Darmois 1953) and Rao in the 1960 (Kagan et al, 1973) concerning the characterization of random variables in linear structures. Only recently this has been recognized by ICA

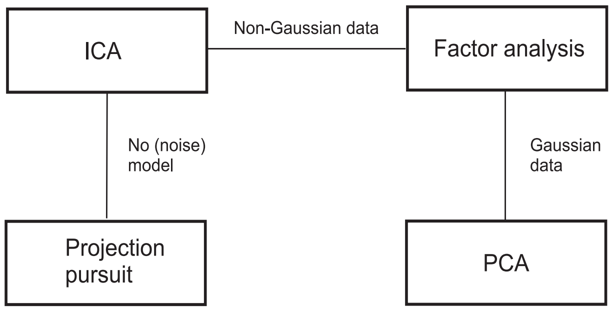


Figure 4.22: (The lines show close connections, and the text next to the lines show the assumptions needed for the connection.)

researchers (Common, 1994, Jutten and Taleb 2000, Fiori 2003) which regrets that this may have delayed progress.

For the general problem of source separation we use the statistical latent variables model. Suppose we have p linear mixtures of m independent components:

$$\begin{bmatrix} x_{i1} \\ \dots \\ x_{ip} \end{bmatrix} = \begin{bmatrix} a_{11}.s_{i1} + \dots + a_{1m}.s_{im} \\ \dots \\ a_{p1}.s_{i1} + \dots + a_{pm}.s_{im} \end{bmatrix} = \begin{bmatrix} a_{11} & \dots & a_{1m} \\ & & \\ a_{p1} & \dots & a_{pm} \end{bmatrix} \begin{bmatrix} s_{i1} \\ \dots \\ s_{im} \end{bmatrix}$$

$$\boldsymbol{x}_i = \boldsymbol{A}\boldsymbol{s}_i \tag{4.2.18}$$

Actually the more general case of unmixing consider a nonlinear transformation and an additional error term ie:

$$\boldsymbol{x}_i = f(\boldsymbol{s}_i) + \boldsymbol{\epsilon}_i \tag{4.2.19}$$

the model 4.2.18 is the base for blind source separation method. Blind means that we know very little or anything on the mixing matrix and make few assumptions on the sources s_i.

The principles behind the techniques of PCA and FA is to limit the number of components s_i to be small that is $m < p$ and is based on obtaining non-correlated sources.

ICA consider the case of $p = m$ and obtain estimates of $\boldsymbol{A}$ and to find components s_i that are independent as possible.

After estimating the matrix $\boldsymbol{A}$, we can compute its inverse, and obtain the independent component simply by:

$$\boldsymbol{s} = \boldsymbol{W}\boldsymbol{x} \tag{4.2.20}$$

Some of the characteristics of the procedure is:

– We fix the variances of the components to be one.

– We cannot determine the order of the components.

– To estimate $\boldsymbol{A}$ at most one of the components can be Gaussian. The independent components must be Gaussian.

The key to estimating the ICA model is non-Gaussianity and independence. There are several methods: maximum likelihood and network entropy, mutual information and Kulback-Leibler divergence, non-linear cross correlations, non-linear PCA, higher-order cumulants, weighted covariance matrix (Hyvarinen, 1999).

Here we outline the Infomax principle of network entropy maximization which is equivalent to maximum likelihood (see Hyvarinen and Oja, 2000). Here we follow Jones and Porril (1998).

Consider eq.4.2.20, $\boldsymbol{s} = \boldsymbol{W}\boldsymbol{x}$. If a signal or source s has a cdf cumulative density function g, then the distribution of $g(s)$ by the probability integral transformation, has an uniform distribution ie has maximum entropy.

The unmixing W can be found by maximizing the entropy of $H(U)$ of the joint distribution $U = (U_1, \ldots, Up) = (g_1(s_1) \ldots g_p(s_p)$ where $s_i = \boldsymbol{W}\boldsymbol{x}_i$. The correct g_i have the same form as the cdf of the $\boldsymbol{x}_i$, and which is sufficient to approximate these cdfs by sigmoids:

$$U_i = tanh(s_i) \tag{4.2.21}$$

Given that $U = g(\boldsymbol{W}\boldsymbol{x})$, the entropy $\boldsymbol{H}(\boldsymbol{U})$ is related to $\boldsymbol{H}(x)$ by:

$$H(\boldsymbol{U}) = H(\boldsymbol{x}) + E(log|\frac{\partial U}{\partial \boldsymbol{x}}|) \tag{4.2.22}$$

Given that we wish to find $\boldsymbol{W}$ that maximizes $\boldsymbol{H}(\boldsymbol{U})$, we can ignore $\boldsymbol{H}(\boldsymbol{x})$ in (4.2.22). Now

$$\frac{\partial U}{\partial \boldsymbol{x}} = |\frac{\partial U}{\partial \boldsymbol{s}}||\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{x}}| = \prod_{i=1}^{p} g^{\text{'}}(s_i)|\boldsymbol{W}| \tag{4.2.23}$$

Therefore 4.2.22 becomes:

$$H(U) = H(\boldsymbol{x}) + E(\sum_{i=1}^{p} g(s_i^{\text{'}})) + log|\boldsymbol{W}| \tag{4.2.24}$$

The term $E = (\sum_{i=1}^{p} g(s_i))$, given a sample of size n, can be estimated by:

$$E(\sum_{i=1}^{p} g(s_i)) \approx \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{p} logg_i^{\text{'}}(s_i^{(j)}) \tag{4.2.25}$$

Ignoring $H(\boldsymbol{x})$, equation 4.2.24 yield a new function which differ from $H(U)$ by $H(\boldsymbol{x})$.

$$h(\boldsymbol{W}) = \frac{1}{n}\sum_{j}^{n}\sum_{i}^{p} log g_i^{,}(s_i^*) + log|\boldsymbol{W}| \tag{4.2.26}$$

Defining the cdf $g_i = tanh$ and recalling that $g^{,} = (1-g^2)$ we obtain:

$$h(\boldsymbol{W}) = \frac{1}{n}\sum_{j=1} n \sum_{i=1}^{p} log(1 - s_i^{,2}) + log|\boldsymbol{W}| \tag{4.2.27}$$

whose maximization with respect to $\boldsymbol{W}$ yield:

$$\bigtriangledown\boldsymbol{W} = \frac{\partial H(\boldsymbol{W})}{\partial \boldsymbol{W}} = \frac{\partial h((W))}{\partial (W)} = (\boldsymbol{W}^{,})^{-1} + 2.\boldsymbol{s}.\boldsymbol{x}^{,} \tag{4.2.28}$$

and an unmixing matrix can be found by taking small steps of size η to $\boldsymbol{W}$

$$\Delta\boldsymbol{W} = \eta((\boldsymbol{W}^{,-1} - 2.\boldsymbol{s}.\boldsymbol{x}^{,}) \tag{4.2.29}$$

After reescaling, it can be shown (see Lee, 1998 p 41,45) that a more general expression is:

$$\Delta\boldsymbol{W} \approx \begin{cases} (\boldsymbol{I} - tanh(\boldsymbol{s})\boldsymbol{s}^{,} - \boldsymbol{s}\boldsymbol{s}^{,})\boldsymbol{W} & \text{super-Gaussian} \\ (\boldsymbol{I} + tanh(\boldsymbol{s})\boldsymbol{s}^{,} - \boldsymbol{s}\boldsymbol{s}^{,})\boldsymbol{W} & \text{sub-Gaussian} \end{cases} \tag{4.2.30}$$

Super-gaussian random variables have typically a "spiky" pdf with heavy tails, ie the pdf is relatively large at zero and at large values of the variable, while being small for intermediate values. A typical example is the Laplace (or double-exponential) distribution. Sub-Gaussian random variables, on the other hand, have typically a "flat" pdf which is rather constant near zero, and very small for large values of the variables. A typical example is uniform distribution.

We end this section with two illustrations of the applications of ICA and PCA to simulated examples from Lee (1998, p 32). The first simulated example involves two uniformly distributed sources s_1 e s_2.

The sources are linearly mixed by:

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} \tag{4.2.31}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$

Figure 4.23 shows the results of applying of the mixing transformation and the application of PCA and ICA. We see that PCA only sphere (decorrelate) the data,

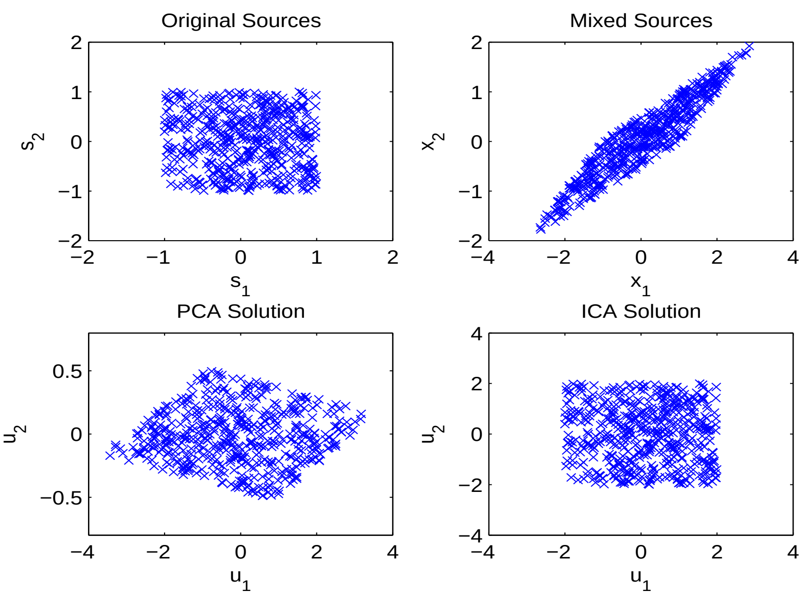


Figure 4.23: PCA and ICA transformation of uniform sources (Top-left:scatter plot of the original sources, Top-right: the mixtures, Bottom-left: the recovered sources using PCA, Bottom-right: the recovered sources using ICA.)

while ICA not only sphere the data but also rotate it such that $\hat{s}_1 = u_1$ and $\hat{s}_2 = u_2$ have the same directions of s_1 and s_2.

The second example in Figure 4.24 shows the time series of two speach signals s_1 and s_2. The signal are linearly mixed as the previous example in eq. 4.2.31. The solution indicates that the recovered sources are permutated and scaled.

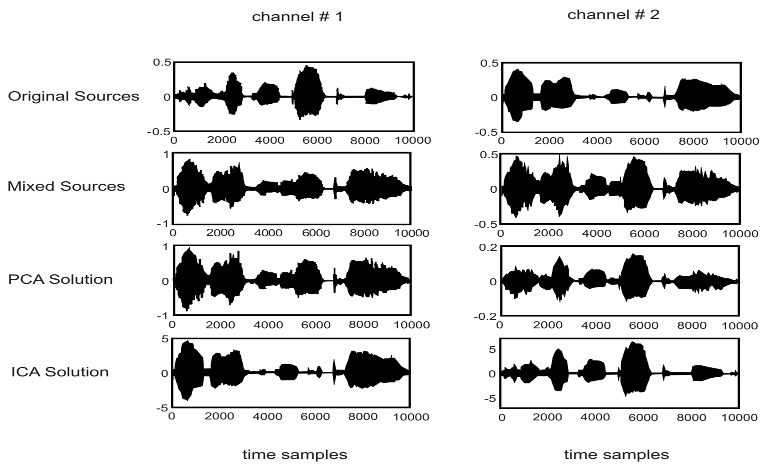


Figure 4.24: PCA and ICA transformation of speach signals (Top: the original sources, second row: the mixtures, third row: the recovere sources using PCA, bottom: the recovered sources using ICA.)

4.2.4 PCA networks

There are several neural networks architecture for performing PCA (See Diamantaras and Kung 1996 and Hertz, Kragh and Palmer,1991). They can be:

– Autoassociator type. The network is trained to minimize:

$$\sum_{i=1}^{n}\sum_{k=1}^{p}(y_{ik}(\boldsymbol{x} - x_{ik}))^2 \tag{4.2.32}$$

or

$$||\boldsymbol{x}' - F_1(\boldsymbol{x})|| = ||A'(\boldsymbol{x} - \boldsymbol{\mu})|| \tag{4.2.33}$$

The architecture with two layer and indentity activation function is as Figure 4.25.

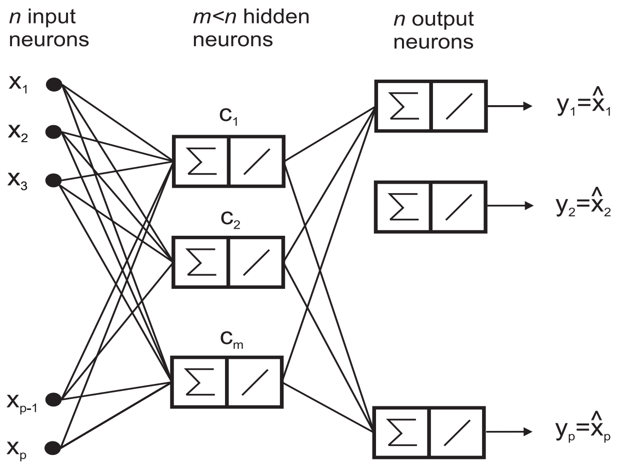


Figure 4.25: PCA networks - Autoassociative

– Networks based on Oja (Hebbian) rule

The network is trained to find $\boldsymbol{W}$ that minimizes

$$\sum_{i=1}^{n}||\boldsymbol{x}_i - \boldsymbol{W}'\boldsymbol{W}\boldsymbol{x}_i||^2 \tag{4.2.34}$$

and its architecture is shown in Figure 4.26.

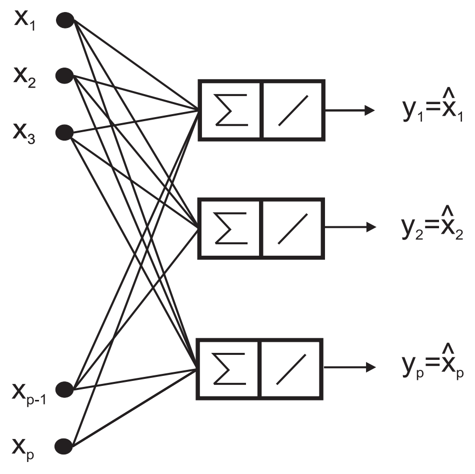


Figure 4.26: PCA networks architecture - Oja rule.

Several update rules has been suggested, all of each follows the same matrix equation:

$$\Delta \boldsymbol{W}_l = \eta_l \boldsymbol{y}_l \boldsymbol{x}_l^{,} - \boldsymbol{K}_l \boldsymbol{W}_l \tag{4.2.35}$$

where $\boldsymbol{x}_l$ and $\boldsymbol{y}_l$ are the input and output vectors, η_l the learning rate and $\boldsymbol{W}_l$ is the weight matrix at the l-th iteration. K_l is a matrix whose choice depends on the specific algorithm, such as the following:

1. William´s rule: $\boldsymbol{K}_l = \boldsymbol{y}_l \boldsymbol{y}_l^{,}$

2. Oja-Karhunen´s rule: $\boldsymbol{K}_l = 3\boldsymbol{D}(\boldsymbol{y}_l \boldsymbol{y}_l^{,}) + 2L(\boldsymbol{y}_n \boldsymbol{y}_n^{,})$

3. Sanger´s rule: $\boldsymbol{K}_l = L(\boldsymbol{y}_l \boldsymbol{y}_l^{,})$

where $D((A))$ is a diagonal matrix with diagonal entries equal to those of $\boldsymbol{A}$, $L(\boldsymbol{A})$ is a matrix whose entries above and including the main diagonal are zero. The other entries are the same as that of $\boldsymbol{A}$. For example:

$$D\begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix} \qquad L\begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 3 & 0 \end{bmatrix} \tag{4.2.36}$$

To illustrate the computation involved consider the $n = 4$ sample of a $p = 3$ variable data in Table 4.8.

Table 4.8: Data $p = 3$, $n = 4$.

n	x_1	x_2	x_3
1	1.3	3.2	3.7
2	1.4	2.8	4.1
3	1.5	3.1	4.6
4	1.1	3.0	4.8

The covariance matrix is:

$$S^2 = \frac{1}{5}\begin{bmatrix} 0.10 & 0.01 & -0.11 \\ 0.01 & 0.10 & -0.10 \\ -0.11 & -0.10 & 0.94 \end{bmatrix} \tag{4.2.37}$$

the eigenvalues and eigenvectors are shown in Table 4.9.

Table 4.9: SVD of the covariance matrix S^2

eigenvalue		eigenvectors		
		$\boldsymbol{V}_1$	$\boldsymbol{V}_2$	$\boldsymbol{V}_3$
d_1	0.965	-0.823	0.553	-0.126
d_2	0.090	-0.542	-0.832	-0.115
d_3	0.084	-0.169	-0.026	-0.985

Now we will compute the first principal component using a neural network architecture.

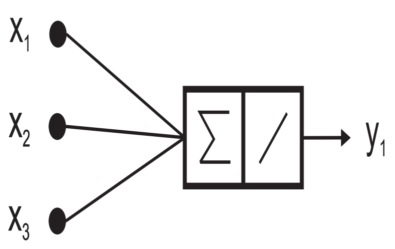


Figure 4.27: Oja´s rule

We choose $\eta_l = 1.0$ for all values of l, and select the initial values $w_1 = 0.3$,$w_2 = 0.4$ and $w_3 = 0.5$. For the first input $\boldsymbol{x}_1 = (0, 0.2, -0.7)$ we have:

$$y = (0.3, 0, 4, 0, 5).(0, 0.2, -0.7) = -0.27 \tag{4.2.38}$$
$$\triangle \boldsymbol{W} = (-0.27(0, 0.2, -0.7) - (-0.27)^2.(0.3, 0.4, 0.5)) \tag{4.2.39}$$
$$\boldsymbol{W} = (0.3, 0.4, 0.5) + \triangle W = (0.278, 0.316, 0.652) \tag{4.2.40}$$

For the next input $\boldsymbol{x}_2 = (0.1, -0.2, -0.3)$

$$y = -0.231 \tag{4.2.41}$$
$$\triangle \boldsymbol{W} = (-0.231(0.10, -0.20, -0.30) - (-0.231)^2.(0.278, 0.316, 0.651)) \tag{4.2.42}$$
$$\boldsymbol{W} = (0.278, 0.316, 0.652) + \triangle W = (0.240, 0.346, 0.687) \tag{4.2.43}$$

Subsequent presentation of $\boldsymbol{x}_3$,$\boldsymbol{x}_4$ and $\boldsymbol{x}_5$ change the weight matrix to $(0.272, 0.351, 0.697)$, $(0.238, 0.313, 0.751)$ and $(0.172, 0.293, 0.804)$, respectively.

This process is repeated, cycling through the input vectors x_1,x_2,. . ., x_5 by the end of the second iteration, the weights becomes $(-0.008, -0.105, 0.989)$ at the end of the third interaction it chances to $(-0.111, -0.028, 1.004)$. The weight adjustment process continues in this manner, resulting in the first principal component.

Further applications are mentioned in Diamantaras and Kung (1996). Most of the applications are in face, vowel, speach recognition. A comparison of the performance of the alternative rules are presented in Diamantaras and Kung (1996) and Nicole (2000). The later author also compare the alternative algorithms for neural computation of PCA with the classical batch SVD using the Fisher´s iris data.

4.2.5 Non-linear PCA networks

The nonlinear PCA networks is trained to minimize:

$$\frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{x}_i - \boldsymbol{W}'\varphi(\boldsymbol{W}\boldsymbol{x})||^2 \tag{4.2.44}$$

where φ is a non-linear function with scalar arguments.

Since from Kolmogorov theorem, a function φ can be approximated by a sum of sinoidals, the architecture of the non-linear PCA network is as in Figure 4.28.

Application on face recognition is given in Diamantaras and Kung (1996).

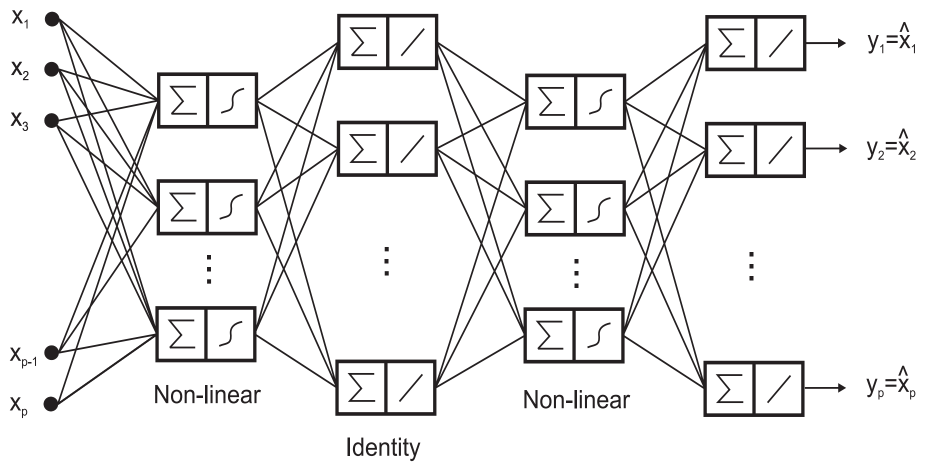


Figure 4.28: Non linear PCA network

4.2.6 FA networks

The application of artificial neural networks to factor analysis has been implemented using two types of architectures:

A) Using autoassociative networks

This network implemented by Sun and Lai (2002) and called SunFA.

The concept consists in representing the correlation matrix $\boldsymbol{R} = (r_{ij})$ by cross-products (or outer product) of two factor symmetric matrices $\boldsymbol{F}_{p\,\mathrm{x}\,m}$ and $\boldsymbol{F}'_{p\,\mathrm{x}\,m}$.

Here the network architecture is that of Figure 4.25 and when there are m common factors the network is trained to minimize the function

$$\begin{aligned} E =&(r_{12} - \sum_{k=1}^{m} f_{1k}f_{2k})^2 + (r_{13} - \sum_{k=1}^{m} f_{1k}f_{3k})^2 + \dots \\ &+ (r_{1p} - \sum_{k=1}^{m} f_{1k}f_{pk})^2 + (r_{23} - \sum_{k=1}^{m} f_{2k}f_{3k})^2 + \dots \\ &(r_{2p} - \sum_{k=1}^{m} f_{2k}f_{pk})^2 + \dots + (r_{p-1,p} - \sum_{k=1}^{m} f_{p-1,k}f_{pk})^2 \end{aligned} \tag{4.2.45}$$

Sun and Lai (2002) report the result of an application to measure ability of 12 students (verbal,numerical,etc) and a simulation result comparing their method to other four methods (factor analysis, principal axes, maximum likelihood and image analysis).

Five correlation matrices were used for samples of size $n = 82{,}164$ and 328 with factors matrices $\boldsymbol{A}$,$\boldsymbol{B}$ an $\boldsymbol{C}$. Ten sets of correlation submatrices where sampled from each correlation matrix. They found that SunFA was the best method.

B) Using PCA (Hebbian) networks

Here an PCA network is used to extract the principal components of the correlation matrix and to choose the number of factors (components) m that will be used in the factor analysis. Then a second PCA network extract m factors choosen in the previous analysis.

This procedure has been used by Delichere and Demmi (2002a,b) and Calvo (1997). Delichere and Demmi use an updating algorithm call GHA (Generalized Hebbian Algorithm) and Calvo the APEX (Adaptive Principal Component Extraction) see Diamantaras and Kung (1996).

We outline Calvo´s application. The data consisted on school grades of 8 students in four subjects (Mathematics, Biology, French and Latin). The first PCA network shown that the first component explained 73% of the total variance and the second explained 27%.

The second network used the architecture shown in Figure 4.29.

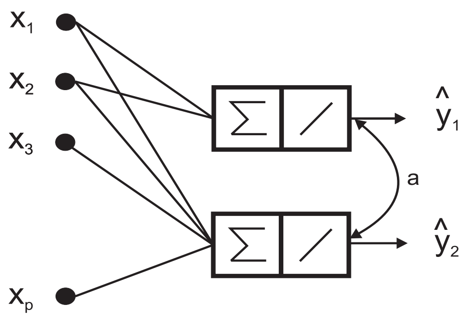


Figure 4.29: Factor analysis APEX network

Calvo obtained the results in Tables 4.10 and 4.11 and Figures 4.2.6 and 4.31, he also reported that the eigenvalues were equal within (± 0.01) to the results obtained on commercial software (SPPS).

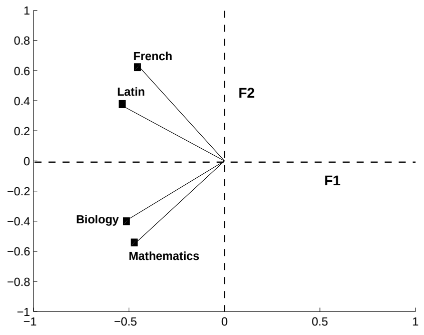


Figure 4.30: Original variables in factor space

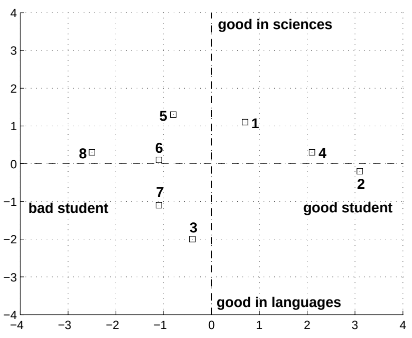


Figure 4.31: Students in factor space

Student	Math	Biology	French	Latin
1	13	12.5	8.5	9.5
2	14.5	14.5	15.5	15
3	5.5	7	14	11.5
4	14	14	12	12.5
5	11	10	5.5	7
6	8	8	8	8
7	6	7	11	9.5
8	6	6	5	5.5

Table 4.10: Scholl grades in four subjects

	Factor 1	Factor 2
Mathematics	-0.4759	-0.5554
Biology	-0.5284	-0.4059
French	-0.4477	0.6298
Latin	-0.5431	0.3647

Table 4.11: Loadings for the two factors retained

4.2.7 Correspondence Analysis Networks - CA

The only reference outlining how neural networks can be used for CA is Lehart (1997). The obvious architecture is the same for SVD or PCA shown in Figure 4.32. Here we take:

$$Z_{ij} = \frac{f_{ij} - f_{i\bullet} f_{\bullet j}}{\sqrt{f_{i\bullet} f_{\bullet j}}} \tag{4.2.46}$$

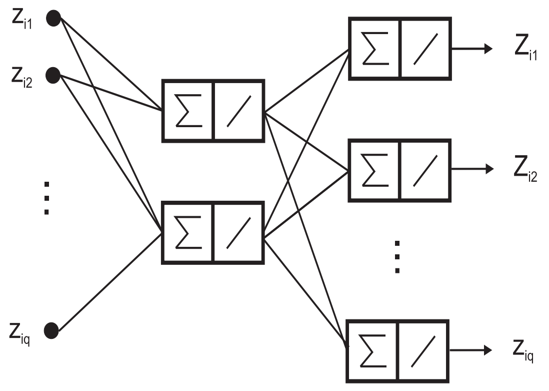


Figure 4.32: CA network

Unfortunately there does not seem to be any application on real data yet.

4.2.8 Independent Component Analysis Networks - ICA

The usual procedure in ICA follows the following steps:

– standardize the data

– whitening (decorrelate i.e apply PCA)

– obtain the independent components (eq. Informax, ie NPCA).

One possible architecture is shown in Figure 4.33 where

$$\boldsymbol{x} = \boldsymbol{Q}\boldsymbol{s} + \boldsymbol{\varepsilon}$$

and the steps are:

– whitening: $\boldsymbol{v}_k = \boldsymbol{V}\boldsymbol{x}_k$

– independent components: $y_k = \boldsymbol{W}v_k$.

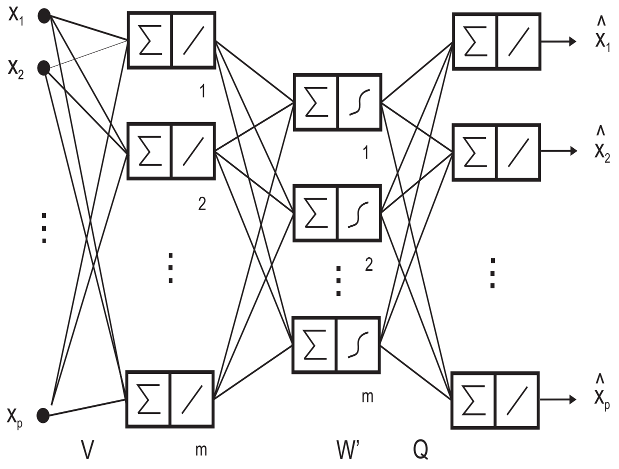


Figure 4.33: ICA neural network

Now, we present some recent applications of ICA.

i) Fisher iris´s data - Lee (1998) compare classification algorithms with a classification based on ICA. The data set contains 3 classes, 4 variables and 50 observations in each class, each class refers to a type of iris plant. The training set contained 75% of the observation and the testing set 25%. The classification error in the test data was 3%. A simple classification with boosting and a k-means clustering gave errors of 4,8% and 4,12% respectively. Figure 4.34 presents the independent directions (basis vectors).

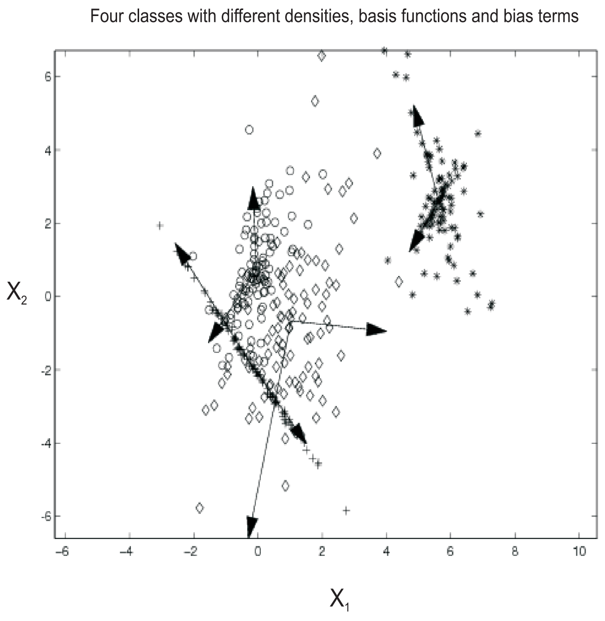


Figure 4.34: An example of classification of a mixture of independent components. There are 4 different classes, each generated by two independent variables and bias terms. The algorithm is able to find the independent directions (basis vectors) and bias terms for each class.

Draghici et al (2003) implement a data mining technique based on the method of ICA to generate reliable independent data sets for different HIV therapies. They consider a coupled model that incorporate three algorithms:
i) ICA model is used for data refinement and normalization. The model accepts a mixture of CD4+, CD8+ and viral load data to a particular patient and normalizes it with broather data sets obtained from other HIV libraries. The ICA is used to isolate groups of independent patterns embedded in the considered libraries.
ii) Cluster (Kohonen Map) networks are used to select those patterns chosen by ICA algorithm that are close to the considered input data.

These two mechanisms helped to select similar data and also to improve the system by throwing away unrelated data sets.
iii) Finally a non-linear regression model is used to predict future mutations in the CD4+,CD8+ and viral loads.

The author points out a series of advantage of their method over the usual mathematical modelling using a set of simultaneous differential equation which requires an expert to incorporate the dynamic behaviour in the equations.

Financial applications: the following are some illustration mentioned by Hyvarinen et al (2001):

i) Application of ICA as a complementary tool to PCA in a study of stock portfolio, allowing the underline structure of the data to be more readily observed. This can be of help in minimizing the risk in the investment strategy.

ii) ICA was applied to find the fundamental factor common to all stores that affect the cashflow, in the same retail chain. The effect of managerial actions could be analysed.

iii) Time series prediction. ICA has been used to predict time series data by first predicting the common components and then transforming back to the original series.

Further applications in finance are Cha and Chan and Chan and Cha (2001). The book of Girolami (1999) also gives an application of clustering using ICA to the Swiss banknote data on forgeries.

4.3 Classification Networks

As mentioned before two important tasks in pattern recognition are patter classification and cluster or vector quantization. Here we deal with the classification problem, whose task is to allocate an object characterized by a number of measurements into one of several distinct classes. Let an object (process, image, individual, etc) be characterized by the k-dimensional vector $\mathbf{x} = (x_1, \ldots, x_k)$ and let c_1,c_2,. . .,c_L be labels representing each of the classes.

The objective is to construct and algorithm that will use the information contained in the components $x_1,\ldots,x_k$ of the vector $\mathbf{x}$ to allocate this vector to one of the L distinct categories. Typically, the pattern classifier or discrimination function calculates L functions (similarity measures), P_1, $P_2,\ldots,P_L$ and allocates the vectors $\mathbf{x}$ to the class with the highest similarity measures.

Many useful ideas and techniques have been proposed for more than 60 years of research in classification problems. Pioneer works are Fisher (1936) Rao (1949) and reviews and comparisons can be seen in Randys (2001), Asparoukhov (2001) and Bose (2003).

Neural network models that are used for classification have already been presented in previous sections. Figures 4.35 to 4.40 presents a taxonomy and examples of neural classifiers.

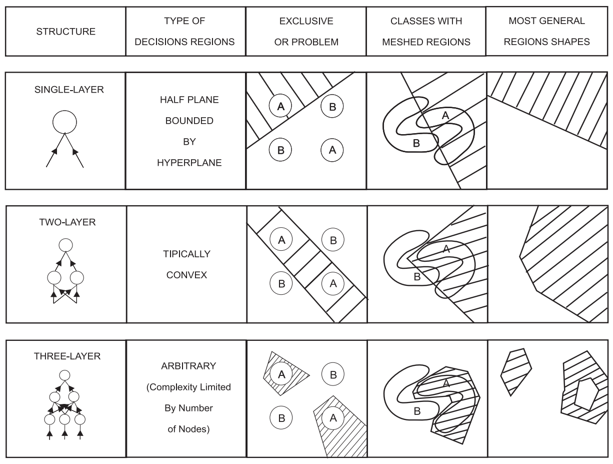

STRUCTURE
TYPE OF DECISIONS REGIONS
EXCLUSIVE OR PROBLEM
CLASSES WITH MESHED REGIONS
MOST GENERAL REGIONS SHAPES
SINGLE-LAYER
HALF PLANE BOUNDED BY HYPERPLANE
TWO-LAYER
TIPICALLY CONVEX
THREE-LAYER
ARBITRARY (Complexity Limited By Number of Nodes)
A
B

Figure 4.35: Taxonomy and decision regions (Type of decisions regions that can be formed by single and multi-layer perceptrons with hard limiting nonlinearities. Shading denotes regions for class A. Smooth closed contours bound input distributions.)

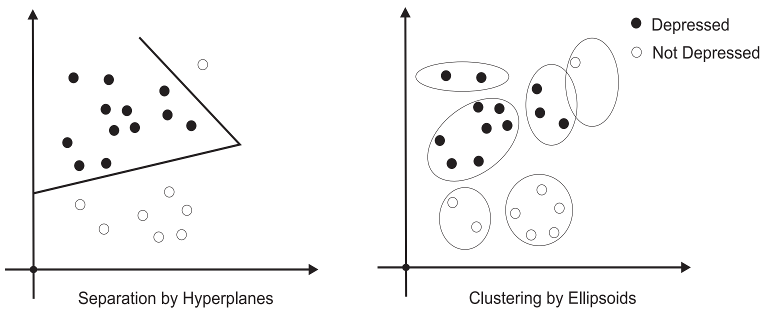


Figure 4.36: Classification by Alternative Networks: multilayer perceptron (left) and radial basis functions networks (right)

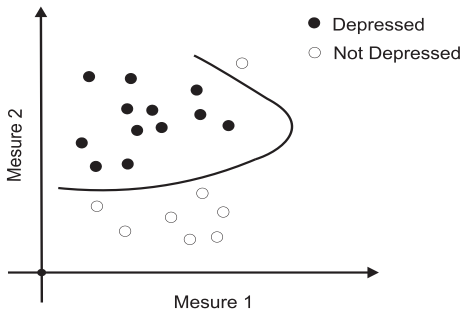


Figure 4.37: A nonlinear classification problem

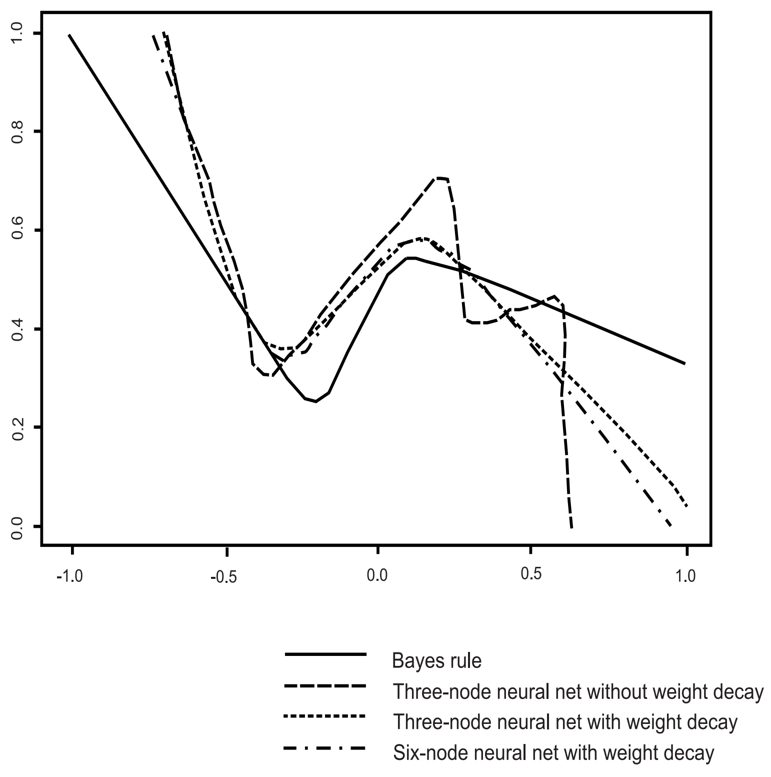


Figure 4.38: Boundaries for neural networks with tree abd six hidden units (note how using weight decay smooths the fitted surface)

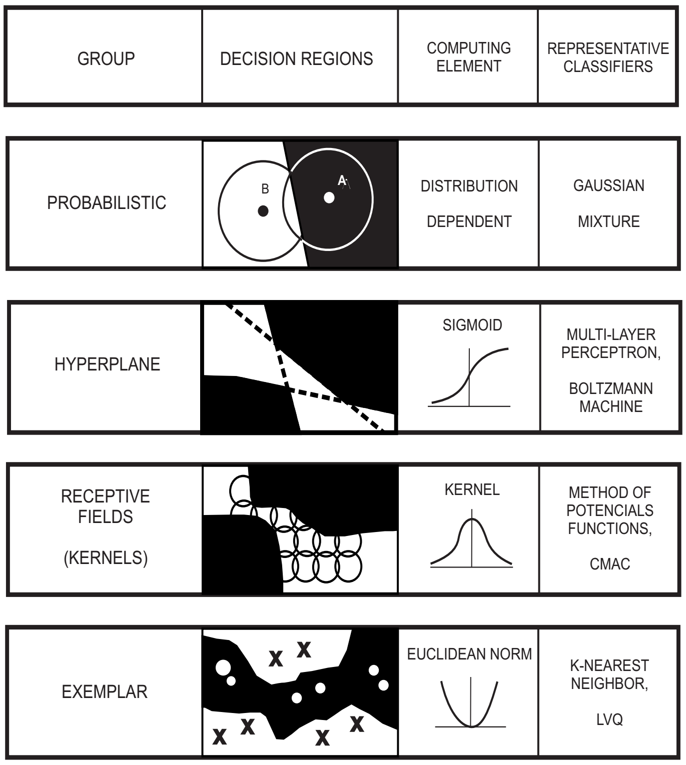


Figure 4.39: Four basic classifier groups

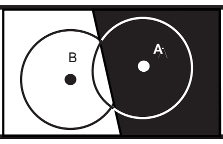
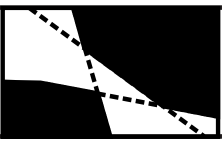
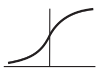
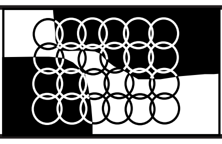
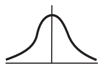
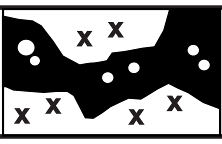
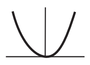
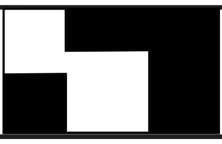
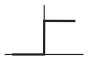
GROUP	DECISION REGIONS	COMPUTING ELEMENT	REPRESENTATIVE CLASSIFIERS
PROBABILISTIC		DISTRIBUTION DEPENDENT	GAUSSIAN, GAUSSIAN MIXTURE
GLOBAL		SIGMOID 	MULTI-LAYER PERCEPTRON, HIGH-ORDER POLYNOMIAL NET
LOCAL		KERNEL 	RADIAL BASIS FUNCTION, KERNEL DISCRIMINANT
NEAREST NEIGHBOR		EUCLIDEAN NORM 	K-NEAREST NEIGHBOR, LEARNING VECTOR QUANTIZER
RULE FORMING		THRESHOLD LOGIC 	BINARY DECISION TREE, HYPERSPHERE

Figure 4.40: A taxonomy of classifiers

Here we give some examples of classification problems solved using neural networks.

We start with an application of PNN - probabilistic neural networks in bankruptcy prediction. Yang at al (1999) using data from 122 companies for the period 1984 to 1989 built on warning bankrupt model for the US oil and gas industry. The data consist of five ratios to classify oil and gas companies into bankrupt and non-bankrupt groups. The five ratios are net cash flow to total assets, total debt to total assets, exploration expenses to total reserves, current abilities to total reserves, and the trend in total reserves calculated on the ratio of change from year to year. The first fou rates are deflated. There are two separate data sets one with deflated ratios and one without deflation.

The 122 companies are randomly divided into three sets: the training data set (33 nonbankrupt companies and 11 bankrupt companies), the validation data set (26 nonbankrupt companies and 14 bankrupt companies) and the testing data (30 nonbankrupt companies and 8 bankrupt companies).

Four methods of classification were tested: FISHER denotes Fisher discriminant analysis, FF denotes feedforward network, PNN means probabilistic neural networks, PNN* is probabilistic neural networks without data normalized. The normalization of data is in the form: $\mathbf{x}^* = \mathbf{x}/||\mathbf{x}||$.

Table 4.12 presents the results and we can see that the ranking of the classification model is: FISHER, PNN*, PNN and EF.

Table 4.12: Number and percentage of Companies Correctly Classified

	Nondeflated data			Deflated data		
	Overall n=38	Nonbankrupt n=30	Bankrupt n=8	Overall n=38	Nonbankrupt n=30	Bankrupt n=8
FISHER	27 (71 %)	20 (67 %)	7 (88 %)	33 (87 %)	26 (87 %)	7 (88 %)
BP	28 (74 %)	24 (80 %)	4 (50 %)	30 (79 %)	30 (100 %)	0 (0 %)
PNN	25 (66 %)	24 (80 %)	1 (13 %)	26 (68 %)	24 (80 %)	2 (25 %)
PNN*	28 (74 %)	24 (80 %)	4 (50 %)	32 (84 %)	27 (90 %)	5 (63 %)

Bounds and Lloyd (1990) compared feedforward network, radial basis functions, three group of clinicians and some statistical classification methods in the diagnostic of low back disorders. The data set referred to 200 patients with low back disorders.

Low back disorder were classified in four diagnostic category: SLBP (simple low backpain), ROOTP (nerve root compression), SPARTH (spinal pathology, due to tumour, inflammation or infection), AIB (abnormal illness behaviour, with significant psychological overlay).

For each patient the data was collected in a tick sheet that listed all the relevant clinical features. The pacient data was analysed using two different subset of the total data. The first data set (full assessment) contained all 145 tick sheet entries. This corresponds to all information including special investigations. The second data used a subset of 86 tick sheet entries for each patient. These entries correspond to a very limited set of symptoms that can be collected by paramedic personnel.

There were 50 patients for each of the four diagnostic class. Of the 50 patients in each class 25 were selected for the training data and the remaining for the test set.

The neural network used were: feedforward network for individual diagnostic ($y = 1$) and ($y = 0$) for remaining three, feedforward with two output (for the four class diagnostic), a network of individual networks with two outputs, and mean of 10 feedforward networks runs after training. All this networks had one hidden layer.

The radial basis networks were designed to recognize all four diagnostic classes. The effect of the number of centers, positions of the centers and different choice of basis functions were all explored.

Two other classification methods and CCAD (computed-aided diagnosis) system were also tested in these data. A summary of the results is shown in Table 4.13 and 4.14. The performance of the radial basis function, CCAD and MLP is quite good.

Table 4.13: Symptomatic assessment: Percentage test diagnoses correct

Classification Method	Simple low back pain	Root pain	Spinal pathology	Abnormal illness behaviour	Mean
Clinicians Bristol neurosurgeons	60	72	70	84	71
Glasgow orthopaedic surgeons	80	80	68	76	76
Bristol general pratictioners	60	60	72	80	68
CCAD	60	92	80	96	82
MLP					
Best 50-20-2	52	92	88	100	83
Mean 50-20-2	43	91	87	98	80
Best 50-0-2	44	92	88	96	80
Mean 50-0-2	41	89	84	96	77
Radial basis functions	60	88	80	96	81
Closest class mean (CCM) (Euclidean)					
50 Inputs	56	92	88	96	63
86 Inputs	68	88	84	96	84
Closest class mean (CCM) (scalar product)					
50 Inputs (K=22)	16	96	44	96	63
86 Inputs (K=4)	92	84	60	84	80
K Nearest Neighboor (KNN) (Euclidean)					
50 Inputs (K=22)	88	84	76	84	83
86 Inputs (K=4)	92	80	84	80	84
K Nearest Neighboor (KNN) (scalar product)					
50 Inputs (K=3)	24	100	56	92	68
86 Inputs (K=6)	100	84	56	84	81

Table 4.14: Full assessment: Percentage test diagnoses correct

Classification Method	Simple low back pain	Root pain	Spinal pathology	Abnormal illness behaviour	Mean
Clinicians Bristol neurosurgeons	96	92	60	80	82
Glasgow orthopaedic surgeons	88	88	80	80	84
Bristol general pratictioners	76	92	64	92	81
CCAD	100	92	80	88	90
MLP					
Best 50-20-2	76	96	92	96	90
Mean 50-20-2	63	90	87	95	83
Best 50-0-2	76	92	88	96	88
Mean 50-0-2	59	88	85	96	82
Radial basis functions	76	92	92	96	89
Closest class mean (CCM) (Euclidean)					
85 Inputs	100	92	72	88	88
145 Inputs	100	88	76	88	88
Closet class mean (CCM) (scalar product)					
85 Inputs (K=22)	12	92	48	100	63
145 Inputs (K=4)	100	72	68	80	80
K Nearest Neighbour (KNN) (Euclidean)					
85 Inputs (K=19)	100	84	80	80	86
145 Inputs (K=4)	96	88	80	80	86
K Nearest Neighbour (KNN) (scalar product)					
85 Inputs (K=19)	48	96	52	92	68
145 Inputs (K=4)	92	92	72	76	83

Santos et al (2005,2006) uses feedforward network in the diagnostic of Hepatitis A and Pulmonary Tuberculosis and gives a comparison with logistic regression.

The hepatitis A data (Santos et al 2005), 2815 individuals from a survey sample from regions in Rio de Janeiro state. The soroprevalence of hepatitis was 36,6. From the 66 collected variables in the study, seven variables reflecting information on the individuals, housing environment, and socioeconomic factors.

For a test sample of 762 individual randomly chosen. Table 4.15 presents the results for the multilayer perceptron and a logistic regression model:

Table 4.15: Sensitivity (true positive) and specificity (true negative)

Errors	Model	
	MLP	LR
Sensitivity (%)	70	52
Specificity (%)	99	99

The Pulmonary Tuberculosis data (Santos et al 2006) consisted of data on 136 patients of the Hospital Universitario (HUCFF - Universidade Federal do Rio de Janeiro). There were 23 variables in total. The test sample had 45 patients. A comparison was made using multilayer perceptron and CART (Classification and Regression Tree) model. Table 4.16 summarizes the results in the test data.

Table 4.16: Sensitivity x Specificity

Errors	Model	
	MLP	CART
Sensitivity (%)	73	40
Specificity (%)	67	70

A general comparison of classification methods including neural networks using medical binary data is given in Asparoukhov and Krzanowski (2001). Table 4.17 reproduces one of their three tables to indicate the methods used (They also have a similar table for large (15-17) and moderate (10) number of variables).

They suggest that:

– all the effective classifiers for large sample are traditional ones (IBM - Independent Binary Model Behaviour, LDF - Linear Discriminating Function, LLM - 2 order loglinear model).

Table 4.17: Leave-one-out error rate (multiplied by 10^3 for footnotesize (6) number of variables[1]

Discriminant procedure	Data set									
	Psychology		Pulmonary		Thrombosis		Epilepsy		Aneurysm	
	$=$	$\neq$	$=$	$\neq$	$=$	$\neq$	$=$	$\neq$	$=$	$\neq$
Linear classifiers										
IBM	239	283	147	139	228	284	168	186	23	21
LDF	244	223	147	139	272	275	147	147	53	46
LD	229	266	147	167	302	343	193	186	28	25
MIP		-		139		325		101		23
Quadratic classifiers										
Bahadur(2)	219	201	152	139	243	343	197	155	23	21
LLM (2)	206	207	151	144	250	304	153	116	23	21
QDF	215	207	-	-	265	314	153	178	-	-
Nearest neighboor classifiers										
kNN-Hills(L)	273(2)	266(1)	187(1)	185(1)	265(1)	294(1)	153(2)	217(1)	363(1)	306(1)
kNN-Hall(L)	230(2)	217(2)	166(1)	144(1)	257(1)	324(1)	102(2)	124(1)	23(1)	21(1)
Other non-parametric classifiers										
Kernel	247	234	166	144	243	363	143	124	23	21
Fourier	271	223	166	154	316	373	177	147	23	21
MLP(3)		207		139		284		132		25
LVQ(c)		190(6)		146(6)		226(6)		109(6)		37(4)

[1]$=$ - equal *prior* probabilities;

$\neq$ = - *prior* probabilities are equal to the class individuals' number divided by the design set size;

kNN-Hills(L) and lNN-Hall(L) - L is order of the procedure;

MLP(h)- h is hidden layer neuron number;

LVQ(c)- c is number of codebooks per class.

– most of the effective classifiers for footnotesize samples are non traditional ones (MLP - Multilayer Perceptron, LVQ - Learning Vector Quantization Neural Network, MIP - Mixed Integer Programming).

Arminger et al (1987) compared logistic discrimination regression tree and neural networks in the analysis of credit risk. The dependent variable is whether the credit is paid off or not. The predictor variable are sex, true of employment, marital status, and car and telephone holder. The result on the test data is shown in Table 4.18.

Table 4.18: Correct Classification - Credit data

	MLP	LR	CART
Good Credit	53	66	65
Bad Credit	70	68	66
Overall	66	67	66

For the lender point of view MLP is more effective.

Desai et al (1997) explore the ability of feedforward network, mixture-of-expert networks, and linear discriminant analysis in building credit scoring models in an credit union environmental. Using data from three credit union data, split randomly in 10 datasets. The data for each credit union contain 962, 918 and 853 observation. One third of each were used as test. The results for the generic models (all data) and the customized models for each credit union is presented in Table 4.19 and 4.20 respectively.

Table 4.19: Generic models

Data set	Percentage correctly classified							
	mlp_g		mmn_g		lda_g		lr_g	
	% total	% bad	% total	% bad	% total	% bad	% total	% bad
1	79.97	28.29	80.12	28.95	79.80	27.63	80.30	33.55
2	82.96	35.06	81.80	38.31	80.60	31.82	81.40	35.06
3	81.04	36.44	79.20	47.46	82.40	37.29	82.70	40.68
4	82.88	33.80	83.33	38.03	83.49	40.85	83.49	44.37
5	79.21	32.88	78.59	43.15	80.60	37.67	80.01	39.04
6	81.04	53.69	80.73	35.57	80.58	38.93	81.35	42.95
7	80.73	44.30	79.82	44.97	80.73	36.24	82.57	41.61
8	78.89	50.00	79.82	41.96	80.58	32.88	81.35	39.73
9	81.92	43.87	80.43	32.90	81.04	30.97	81.35	33.55
10	79.51	62.50	80.73	32.64	81.35	33.33	82.42	40.28
average	80.75	42.08	80.46	38.39	81.12	34.76	81.70	39.08
p-value[1]			0.191	0.197	0.98	0.035	0.83	0.19

[1] The p-values are for one-tailed paired t-test comparing mlp_g results with the other three methods.

The results indicates that the mixture-of-experts performns better in classifying bad credits.

Table 4.20: Customized models

Data set	Sample	Percentage correctly classified					
		mlp_c		lda_c		lr_c	
		% total	% bad	% total	% bad	% total	% bad
Credit union L:							
1	14.88	83.33	32.00	83.30	16.00	82.70	24.00
2	17.26	87.50	31.03	82.70	20.69	81.00	34.48
3	17.26	86.90	41.38	83.90	31.03	82.10	37.93
4	22.02	81.55	37.84	84.52	32.43	82.74	37.84
5	18.45	82.14	29.03	77.40	16.13	78.00	38.71
6	17.26	83.33	41.38	81.55	27.59	82.74	41.38
7	21.43	80.36	16.67	78.50	08.33	80.95	30.56
8	17.86	79.76	56.67	82.14	10.00	81.55	23.33
9	16.67	85.71	32.14	86.31	32.14	86.90	39.29
10	18.45	83.33	29.03	79.76	19.35	80.36	29.03
Credit union M:							
1	26.77	85.43	79.71	86.60	70.97	87.40	73.53
2	26.77	88.58	76.47	85.40	18.64	88.20	79.41
3	20.47	85.43	80.77	87.40	31.58	85.80	75.00
4	23.63	90.94	75.00	90.20	35.14	89.00	75.00
5	24.02	89.37	88.52	89.90	22.22	86.60	81.97
6	26.38	88.58	74.63	88.19	12.96	86.22	71.64
7	26.77	85.43	74.63	85.04	28.30	86.22	77.61
8	25.59	88.19	75.38	86.61	28.89	87.40	67.69
9	27.59	87.40	72.86	85.43	31.37	86.61	67.14
10	24.41	90.55	82.26	89.37	21.05	89.37	80.65
Credit union N:							
1	25.43	75.86	49.15	75.40	18.64	78.50	27.11
2	24.57	77.15	40.35	??	31.58	75.50	24.56
3	15.95	81.90	35.14	83.20	35.14	84.10	35.14
4	19.40	77.15	44.44	78.90	22.22	80.60	28.89
5	23.28	76.72	24.07	75.40	12.96	75.90	18.52
6	22.84	79.31	35.85	75.00	28.30	76.22	26.42
7	19.40	81.90	31.11	80.60	28.89	81.03	28.89
8	21.98	74.13	37.25	74.57	31.37	75.43	31.37
9	24.57	80.17	47.37	77.59	21.05	78.88	21.05
10	21.98	77.59	19.61	77.16	21.57	76.72	19.61
average		83.19	49.72	82.35	38.49	82.67	44.93
p-value[1]				0.018	5.7E-7	0.109	0.0007

[1] The p-values are for one-tailed paired t-test comparing mlp_c results with the other two methods.

West (2000) investigate the credit scoring accuracy of five neural network models: multilayer perceptron (MLP), mixture-of-experts, radial basis function, learning vector quantization and fuzzy adaptive resonance. The neural networks credit scoring models are tested using 10-fold crossvalidation with two real data sets. Results are compared with linear discriminant analysis, logistic regression, k-nearest neighbour, kernel density estimation and decision trees

Results show that feedforward network may not be the most accurate neural network model and that both mixture of experts and radial basis function should be considered for credit scoring applications. Logistic regression is the most accurate of the traditional methods.

Table 4.21 summarize the results.

Table 4.21: Credit scoring error, average case neural network models

	German credit data[b]			Australian credit data[b]		
	Good credit	Bad credit	Overall	Good credit	Bad credit	Overall
Neural models[a]						
MOE	0.1428	0.4775	0.2434	0.1457	0.1246	0.1332
RBF	0.1347	0.5299	0.2540	0.1315	0.1274	0.1286
MLP	0.1352	0.5753	0.2672	0.1540	0.1326	0.1416
LVQ	0.2493	0.4814	0.3163	0.1710	0.1713	0.1703
FAR	0.4039	0.4883	0.4277	0.2566	0.2388	0.2461
Parametric models						
Linear discriminant	0.2771	0.2667	0.2740	0.0782	0.1906	0.1404
Logistic regression	0.1186	0.5133	0.2370	0.1107	0.1409	0.1275
Non-parametric models						
K nearest neighbor	0.2257	0.5533	0.3240	0.1531	0.1332	0.1420
Kernel density	0.1557	0.6300	0.3080	0.1857	0.1514	0.1666
CART	0.2063	0.5457	0.3044	0.1922	0.1201	0.1562

[a] Neural network results are averages of 10 repetitions.

[b] Reported results are group error rates averaged across 10 independent holdout samples.

Markham and Ragsdale (1995) presents an approach to classification based on combination of linear discrimination function (which the label Mahalanobis distance measure - MDM) and feedforward network. The characteristic of the data set is shown in Table 4.22. The process of splitting the data in validation and test set was repeated 30 times. The results are shown in Table 4.23.

Table 4.22: Data Characteristic

Dataset	1	2
Problem Domain	Oil Quality Rating	Bank Failure Prediction
Number of groups	3	2
Number of observations	56	162
Number of variables	5	19

Finally, a recent reference on comparisons of classification models is Bose (2003).

Table 4.23: Comparison of classification methods on three-group oil quality and two-group bank failure data set

	Percentage of Misclassified Observations in the Validation Sample					
	Three-group oil quality			Two-group bank failure		
Sample	MDM	NN1	NN2	MDM	NN1	NN2
1	22.22	18.52	3.70	10.00	3.75	2.50
2	14.81	18.52	11.11	17.50	10.00	7.50
3	11.11	11.11	0	20.00	11.25	7.50
4	11.11	0	0	15.00	10.00	5.00
5	18.52	22.22	14.81	11.25	6.25	3.75
6	3.70	0	0	11.25	5.00	1.25
7	14.81	0	0	12.50	8.75	5.00
8	29.63	22.22	11.11	13.75	7.50	3.75
9	7.41	0	0	12.50	6.25	5.00
10	18.52	7.41	0	12.50	7.50	2.50
11	3.70	0	0	17.50	8.75	6.25
12	18.52	11.11	0	20.00	11.25	7.50
13	7.41	0	0	15.00	6.25	3.75
14	7.41	0	0	13.75	5.00	5.00
15	11.11	0	0	16.25	6.25	2.50
16	7.41	0	0	18.75	10.00	6.25
17	18.52	7.41	3.70	25.00	13.75	8.75
18	11.11	3.70	0	13.75	10.00	6.25
19	3.70	0	0	16.25	8.75	5.00
20	14.81	7.41	0	13.75	6.25	3.75
21	11.11	0	0	15.00	7.50	3.75
22	14.81	3.70	0	13.75	6.25	2.50
23	25.93	18.52	11.11	15.00	8.75	5.00
24	22.22	11.11	11.11	20.00	10.00	6.25
25	18.52	7.41	0	15.00	11.25	5.00
26	14.81	7.41	0	17.50	8.75	5.00
27	7.41	0	0	18.75	8.75	7.50
28	14.81	3.70	3.70	16.25	5.00	2.50
29	0	0	0	15.00	6.25	3.75
30	14.81	7.41	3.70	16.25	7.50	5.00
Average	13.33	6.42	2.47	15.63	8.08	4.83

Chapter 5

Regression Neural Network Models

5.1 Generalized Linear Model Networks - GLIMN

Generalized (iterative) linear models (GLIM) encompass many of the statistical methods most commonly employed in data analysis. Beyond linear models with normal distributed errors, generalized linear models include logit and probit models for binary response variable and log-linear (Poisson-regression) models for counts.

A generalized linear model consists of three components

- A random component, in the form of a response variable Y with distribution from the exponential family which includes: the normal, Poisson, binomial, gamma, inverse-normal, negative-binomial distributions, etc.

- A linear predictor

$$\eta_i = \alpha_0 + \alpha_1 x_{i1} + \ldots + \alpha_k x_{ik} \tag{5.1.1}$$

  on which y_i depends. The x' are independent (covariates, predictors) variables.

- A link function $L(\mu_i)$ which relates the expectation $\mu_i = E(Y_i)$ to the linear predictor η_i

$$L(\mu_i) = \eta_i \tag{5.1.2}$$

The exponential family of distribution includes many common distributions and its parameters have some nice statistical properties related to them (sufficiency, attain Cramer-Rao bound). Members of this family can be expressed in the general form

$$f(y, \theta, \phi) = \exp\left\{\frac{y\theta - b(\theta)}{\alpha(\phi) + C(y, \phi)}\right\}. \qquad (5.1.3)$$

If ϕ is known, then θ is called the *natural* or *canonical* parameter. When $\alpha(\phi) = \phi$, ϕ is called the *dispersion* or *scale* parameter. It can be shown that

$$E(Y) = b'(\theta) = \mu \quad \mathrm{Var}(Y) = \alpha(\phi)b''(\theta) = \alpha(\phi) \vee (\theta). \qquad (5.1.4)$$

Fitting a model may be regarded as a way of replacing a set of data values $\underset{\sim}{y} = (y_i \dots y_n)$ by a set of fitted values $\underset{\sim i}{\hat{\mu}} = (\hat{\mu}_1, \dots, \hat{\mu}_n)$ derived from a model involving (usually) a relative small number of parameters. One measure of discrepancy most frequently used is that formed by the logarithm of the ratio of likelihoods, called deviance (D).

Table 5.1 gives some usual choices of link functions for some distributions of $\underset{\sim}{y}$ for the canonical link $\theta = \eta$.

Table 5.1: Generalized Linear Models: mean, variance and deviance functions

	η	$\vee(\mu)$	$D(\hat{\mu})$
Normal	μ	1	$\sum(y - \hat{\mu})^2$
Binomial	$\ln[\mu/(1-\mu)]$	$\mu(1-\mu)$	$2\Sigma\left\{[y\ln(y/\hat{\mu})] + \right.$ $\left. +(n-y)\ln[(n-y)/(n-\hat{\mu})]\right\}$
Poisson	$\log\mu$	μ	$2\Sigma[y\ln(y/\hat{\mu}) - (y-\hat{\mu})]$
Gamma	μ^{-1}	μ^2	$2\Sigma\left[-\ln(y/\hat{\mu}) - (y-\hat{\mu})/\hat{\mu}\right]$

The neural network architecture equivalent to the GLIM model is a perceptron with the predictors as inputs and one output. There are no hidden layers and the activation function is chosen to coincide with the inverse of the link function (L). This network is shown in Figure 5.1.

In what follows we will deal with some particular cases: the logit regression and the regression models neural networks.

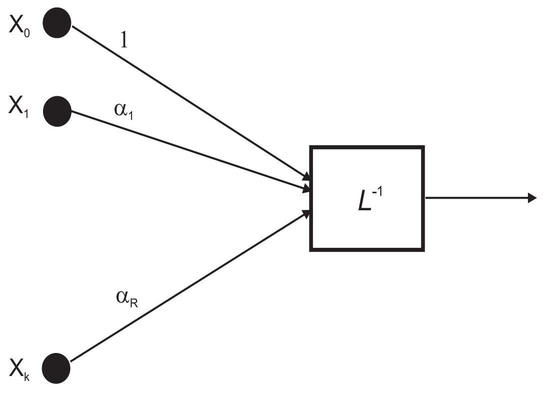


Figure 5.1: GLIM network

5.1.1 Logistic Regression Networks

The logistic model deals with a situation in which we observe a binary response variable y and k covariates x's. The logistic regression model is a generalized linear model with binomial distribution for $y_1 \dots y_n$ and logit link function. The equation takes the form

$$p = P(Y = 1/\underset{\sim}{x}) = \frac{1}{1 + \exp(-\beta_0 - \sum_{i=1}^{k} \beta_i x_i)} = \wedge \left(\beta_0 + \sum_{i=1}^{k} \beta_i x_i \right) \quad (5.1.5)$$

with $\wedge$ denoting the logistic function. The model can be expressed in terms of log odds of observing 1 given covariates x as

$$\log tp = \log \frac{P}{1-\beta} = \beta_0 + \sum_{i=1}^{k} \beta_i x_i. \quad (5.1.6)$$

A natural extension of the linear logistic regression model is to include quadratic terms and multiplicative interaction terms:

$$P = \Lambda \left(\beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \gamma_i x_i^2 + \sum_{i<j} \delta_{ij} x_i x_j \right). \quad (5.1.7)$$

Other approaches to model the probability p include generalized additive models (Section 5.2.3) and feed-forward neural networks.

For the linear logistic model (5.1.6) the neural network architecture is as in Figure 5.1 with the sigmoid link k function as activation function of the output neuron.

A feed-forward network with hidden layer with r neurons and sigmoid activation function would have the form:

$$P = \Lambda\left(w_0 + \sum_{j=1}^{r} w_j \Lambda(w_{oj} + \sum_{j=1}^{k} w_{ij} x_i)\right). \tag{5.1.8}$$

This representation of neural network is the basis for analytical comparison of feed-forward network and logistic regression. For a neural network without a hidden layer and a sigmoid activation function (5.1.8) reduces to (5.1.6) and this network is called logistic perceptron in the neural network literature. Therefore it does not make sense to compare a neural network with hidden layer and linear logistic regression. For a discussion of this and related measures and other comparisons among these two models see Schwarzer et al (2000) and Tu (1996). An attempt to improve the comparison and to use logistic regression as a staring point to design a neural network which would be at least as good as the logistic model is presented in Ciampi and Zhang (2002) using ten medical data sets.

The extension to the polychotomous case is obtained considering $K > 1$ outputs. The neural network of Figure 5.2 is characterized by weights $\beta_{ij}(j = 1, \ldots, K)$ to output y_j, obtained by

$$p_j = \Lambda\left(\beta_{oj} + \sum_{i=1}^{k} \beta_{ij} x_i\right). \tag{5.1.9}$$

In statistical terms this network is equivalent to the multinomial or polychotomous logistic model defined as

$$p_j = P(y_j = 1/\underset{\sim}{x}) = \frac{\Lambda(x, \beta_j)}{\sum_{j=1}^{k} \Lambda(x, \beta_j)} \tag{5.1.10}$$

the weights can be interpreted as regression coefficients and maximum likelihood obtained by back propagation maximum likelihood.

There is a huge literature, specially in medical sciences, with comparisons among these two models (LR and ANN). Here we will only give some useful references mainly from the statistical literature. This section ends with two applications showing some numerical aspects of the models equivalence as given in Schumacher et al (1996).

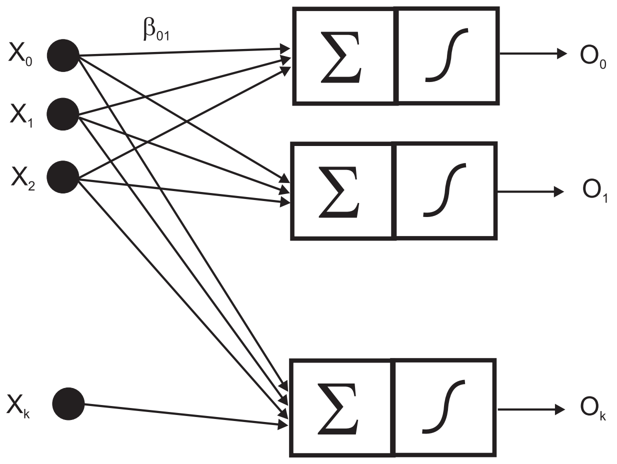


Figure 5.2: Policholomous logistic network

Applications and comparisons of interest are: personal credit risk scores (Arminger et al, 1997), country international credit scores (Cooper, 1999), polychotomous logistic for discrete choice (Carvalho et al, 1998), scores risk of insolvency (Fletcher and Goss, 1993, Leonard et al, 1995, Brockett et al, 1997) and some medical applications (Hammad et al, 1997, Ottenbacher et al, 2001, Nguyen et al, 2002, Boll et al, 2003).

The first illustration is the that of Finney (1947) and the data consists $N = 39$ binary responses denoting the presence $(y = 1)$ or absence $(y = 0)$ of vasoconstriction in the finger's skin after the inspiration of an air volume at a mean rate of inspiration R.

Table 5.2 presents the results of the usual analysis from two logistic regression models. The first, considers only one covariate $x_1 = \log R$, that is

$$P(y = 1/x) = \Lambda(\beta_0 + \beta_1 x_1). \tag{5.1.11}$$

The second model considers the use of a second covariate, that is

$$P(y = /x) = \Lambda(\beta_0 + \beta_1 + \beta_2 x_2). \tag{5.1.12}$$

For $x_1 = \log$ and $x_2 = \log V$, respectively,

In the first case, the results show an effect marginally significant of the logarithm of rate of inspiration in the probability to the presence of vasoconstriction in the finger's skin.

Table 5.2: Results of a logistic regression analysis for the vasoconstriction data

Variable	Coefficient of Regression	Standard Error	P-value	Neural Network
Intercept	-0.470	0.439	-	-0.469
Log R	1.336	0.665	0.045	1.335
		$(L = 24543)$		$(E = 24.554)$
Intercept	-2.924	1.288	-	-2.938
Log R	4.631	1.789	0.010	-4.651
Log V	5.221	1.858	0.005	5.240
		$(L = 14.632)$		$(E = 14,73)$

An ANN equivalent to this model would be a feed-forward network with two input neurons $(x_0 = 1, x_1 = \log R)$. Using a rate of learning $\eta = 0,001$ to the last interactions ML-BP, we have $w_0 = -0,469$ and $w_1 = 1,335$ with a Kullback-Leiber measure of $E^* = 24,554$.

For the case of the model with two covariates, the result shows a significant influence of the two covariates in the probability of vasoconstriction in the finger's skin. One Perceptron with three input neurons $(x0 = 1, x1 = \log R, x2 = \log V)$ should give similar results if ML-BP was used. This was almost obtained $(w_0 = 2,938, w_1 = 4,650, w_2 = 4,240)$ although the algorithm backpropagation demands several changes in learning rate from an initial value $\eta = 0,2$ to $\eta = 0,000001$ involving approximately 20000 interaction to obtain a value of the distance Kullback-Leibler of $E^* = 14,73$ comparable to that of minus the log-likelihood $(L = -14,632)$.

The second example is a study to examine the role of noninvasive sonographic measurements for differentiation of benign and malignant breast tumors. Sonography measurements in N-458 women and characteristics $(x_1, \ldots, x_p)$ were collected.

Cases were verified to be $325(y = 0)$ benign and 133 as $(y = 1)$ malignant tumor. A preliminary analysis with a logistic model indicates three covariates as significant, this indication plus problems of collinearity among the six variables suggests the use of age, number of arteries in the tumor (AT), number of arteries in the contralateral breast (AC). The results are shown in Table 5.3.

A comparison with others ANN is shown in the Table 5.4.

Table 5.3: Results of a logistic regression analysis of breast tumor data

Logistic				
Variable	Coefficients	Standard Error	P Value	Neural Network Weights
Intercept	-8,178	0.924		$w_0 = -8.108$
Age	0.070	0.017	0.0001	$w_1 = 0.069$
log AT+1	5.187	0.575	0.0001	$w_2 = 5.162$
log AC+1	-1.074	0.437	0.0014	$w_3 = -1.081$
		L=79.99		E=80.003

Table 5.4: Results of classification rules based on logistic regression, CART and feed-forward neural networks with j hidden units (NN(J)) for the breast tumor data

Method	Sensitivity (%)	Specificity (%)	Percentage of correct classification
Logistic regression	95.5	90.2	91.7
CART	97.9	89.5	91.9
NN(1)	95.5	92.0	93.0
NN(2)	97.0	92.0	93.4
NN(3)	96.2	92.3	93.4
NN(4)	94.7	93.5	93.9
NN(6)	97.7	94.8	95.6
NN(8)	97.7	95.4	96.1
NN(10)	98.5	98.5	98.5
NN(15)	99.2	98.2	98.5
NN(20)	99.2	99.1	99.1
NN(40)	99.2	99.7	99.6

5.1.2 Regression Network

The regression network is a GLIM model with identity link function. The introduction of hidden layer has no effect in the architecture of the model because of the linear properties and the architecture remains as in Figure 5.1.

Some authors have compared the prediction performance of econometric and regression models with feed-forward networks usually with one hidden layer. In fact, they compared linear models with nonlinear models, in this case a particular projection pursuit model with the sigmoid as the smooth function. (See Section 5.2.5). An extension for system of equations can be obtained for systems of k equations by considering k outputs in the neural network as in Figure 5.2. Again the outputs activation function are the identity function.

In what follows we describe some comparisons of regression models versus neural networks.

Gorr et al (1994) compare linear regression, stepwise polynomial regression and a feed-forward neural network with three units in the hidden layer with an index used for admissions committee for predicting students GPA (Grade Point Average) in professional school. The dependent variable to be predicted for admissions decision is the total GPA of all courses taken in the last two years. The admission/decision is based on the linear decision rule.

$$LDR = math + chem + eng + zool + 10Tot + 2Res + 4PTran \quad (5.1.13)$$

which are the grades in Mathematics, Chemistry, English, Zoology; and total GPA. The other variables are binary variables indicating: residence in the area, a transfer student had partial credits in prerequisites transferred, and a transfer student had all credits in prerequisites transfer.

Although none of the empirical methods was statistically significant better than the admission committee index, the analysis of the weights and output of the hidden units, identifies three different patterns in the data. The results showed obtained very interesting comparisons by using quartiles of the predictions.

Church and Curram (1996) compared forecast of personal expenditure obtained by neural network and econometric models (London Business School, National Institute of Economic and Social Research, Bank of England and a Goldman Sache equation). None of the models could explain the fall in expense growth at the end of the eighties and beginning of the nineties.

A summary of this application of neural networks is the following:

- Neural networks use exactly the same covariates and observations used in each econometric model. These networks produced similar results to the econometric models.

- The neural networks used 10 neurons and a hidden layer, in all models.

- The dependent variable and the covariates were re-escalated to fall in the interval 0,2 to 0,8.
- A final exercise used a neural network with inputs of all variables from all models. This network was able to explain the fall in growth, but it is a network with too many parameters.
- Besides the forecast comparison, a sensitivity analysis was done on each variable, so that the network was tested with the mean value of the data and each variable was varied to verify the grade in which each variation affects the forecast of the dependent variable.

Another econometric model comparison using neural network and linear regression was given by Qi and Maddala (1999). They relate economic factors with the stock market and in their context conclude that neural network model can improve the linear regression model in terms of predictability, but not in terms of profitability.

In medicine, Lapeer et al (1994) compare neural network with linear regression in predicting birthweight from nine perinatal variables which are thought to be related. Results show, that seven of the nine variables, i.e. gestational age, mother's body-mass index (BMI), sex of the baby, mother's height, smoking, parity and gravidity, are related to birthweight. They found that neural network performed slightly better than linear regression and found no significant relation between birthweight and each of the two variables: maternal age and social class.

An application of neural network in fitting response surface is given by Balkin and Lim (2000). Using simulated results for an inverse polynomial they shown that neural network can be a useful model for response surface methodology.

5.2 Nonparametric Regression and Classification Networks

Here we relate some nonparametric statistical methods with the models found in the neural network literature.

5.2.1 Probabilistic Neural Networks

Suppose we have C classes and want to classify a vector $\underset{\sim}{x}$ to one of this classes. The Bayes classifier is based on $p(k/\underset{\sim}{x})\alpha\pi_k p_k(\underset{\sim}{x})$, where π_k is the prior probability of a observation be from class $k, p_k(\underset{\sim}{x})$ is the probability of $\underset{\sim}{x}$ being observed among those of class $C = k$.

In general, the probability π_k and the density $p(\cdot)$ have to be estimated from the data. A powerful nonparametric technique to estimate these functions is based on kernel methods and propose by Parzen (Ripley, 1996, p.184).

A kernel K is a bounded function with integral one. Suitable example is the multivariate normal density function. We use $K(\underset{\sim}{x} - \underset{\sim}{y})$ as a measure of the proximity of $\underset{\sim}{x}$ on $\underset{\sim}{y}$. The empirical distribution of $\underset{\sim}{x}$ within a group k gives mass $\frac{1}{m_k}$ to each of the samples. A local estimate of the density $p_k(\underset{\sim}{x})$ can be found summing all these contributions with weight $K(\underset{\sim}{x} - \underset{\sim}{x}_2)$ that is (Ripley, 1996, p.182)

$$\hat{p}_j(\underset{\sim}{x}) = \frac{1}{n_j}\sum_{i=1}^{n_j} K(\underset{\sim}{x} - \underset{\sim}{x}_i). \tag{5.2.1}$$

This is an average of the kernel functions centered on each example from the class. Then, we have from Bayes theorem:

$$\hat{p}(k/\underset{\sim}{x}) = \frac{\pi_k \hat{p}_k(\underset{\sim}{x})}{\sum_{j=1}^{C} \pi_j \hat{p}_j(\underset{\sim}{x})} = \frac{\frac{\pi_k}{n_k}\sum_{[i]=k} K(x - x_k)}{\sum_i \frac{\pi_{[i]}}{n_{[i]}} K(x - x_i)}. \tag{5.2.2}$$

When the prior probability are estimated by $n_{k/n}$, (5.2.2) simplifies to

$$p(k/\underset{\sim}{x}) = \frac{\sum_{[i]=k} K(\underset{\sim}{x} - \underset{\sim}{x}_i)}{\sum_i K(\underset{\sim}{x} - \underset{\sim}{x}_i)}. \tag{5.2.3}$$

This estimate is known as Parzen estimate and probabilistic neural network (PNN) (Patterson, 1996, p.352).

In radial basis function networks with a Gaussian basis it is assumed that the normal distribution is a good approximation to the cluster distribution. The kernel nonparametric method of Parzen approximates the density function of a particular class in a pattern space by a sum of kernel functions, and one possible kernel is the Gaussian function. As we have seen the probability density function for a class is approximated by the following equation

$$p_k(\underset{\sim}{x}) = \left(\frac{1}{2\pi^{n/2}\sigma^2}\right)\frac{1}{n_K}\sum_{j=1}^{n_k} e^{-(x-x_{kj})^2/2\sigma^2} \tag{5.2.4}$$

where σ^2 has to be decided. Choosing a large value results in overgeneralization, choosing a value too small results in overfitting. The value of σ should be dependent on the number of patters in a class. One function used is

$$\sigma = a n_k^{-b} \; a > 0, 0 < b < 1. \tag{5.2.5}$$

A three layer network like a RBF network is used with each unit in the hidden layer centered on an individual item data. Each unit in the output layer has weights of 1, and a linear output function, this layer adds all the outputs from the hidden layer that corresponds to data from the same class together. This output represents the probability that the input data belongs to the class represented by that unit. The final decision as to what class the data belongs to is simply the unit with the output layer with the largest value. If values are normalized they lie between 0 and 1. In some networks an additional layer is included that makes this decision as a winner takes all, each unit with binary output.

Figure 5.3 shows a network that finds three class of data. There are two input variables, and for the three classes of data there are four samples for class A, three samples for class B and two samples for class C; hence the number of neurons in the hidden layer.

The advantage of the PNN is that there is no training. The values for the weights in the hidden units (i.e. the centers of the Gaussian functions) are just the values themselves. If the amount of data is large, some clustering can be done to reduce the number of units needed in the hidden k layer.

An application of PNN to classification of electrocardiograms with a 46-dimensional vector input pattern, using data of 249 patients for training and 63 patients for testing is mentioned in Specht (1988).

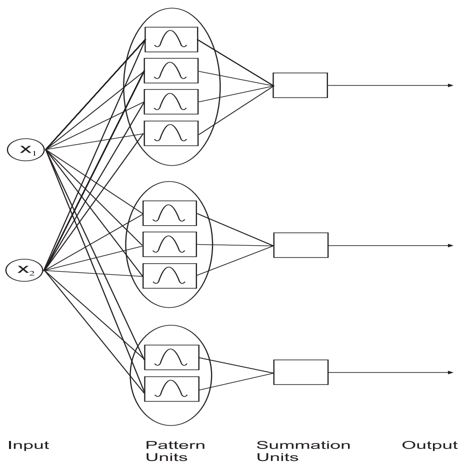


Figure 5.3: Probabilistic Neural Network - PNN

5.2.2 General Regression Neural Networks

The general regression neural network was named by Specht (1991) and its was already known in the nonparametric statistical literature as Nadaraya-Watson estimator.

The idea of GRNN is to approximate a function given a set of n outputs $y_1, \ldots, y_n$ and n vector of inputs $\underset{\sim 1}{x}, \ldots \underset{\sim n}{x}$ using the following equation from probability theory for the regression of y given $\underset{\sim}{x}$

$$\bar{y}(\underset{\sim}{\bar{x}}) = E(y/\underset{\sim}{x}) = \frac{\int y f(\underset{\sim}{x}, y) dy}{\int f(\underset{\sim}{x}, y) dy} \tag{5.2.6}$$

where $f(\underset{\sim}{x}, y)$ is the joint probability density function of $(\underset{\sim}{x}, y)$.

As in PNN we approximate the conditional density function by the sum of Gaussian function. The regression of y on $\underset{\sim}{x}$ is then given by

$$\hat{y}(\underset{\sim}{\hat{x}}) = \frac{\sum_{j=1}^{n} y_j e^{-d_j^2/2\sigma^2}}{\sum_{j=1}^{n} e^{-d_j^2/2\sigma^2}} \tag{5.2.7}$$

a particular form of the Nadaraya-Watson estimator

$$\hat{y}(x) = \frac{\sum y_i K(\underset{\sim}{x} - \underset{\sim i}{x})}{\sum K(\underset{\sim}{x} - \underset{\sim i}{x})}. \tag{5.2.8}$$

The value of d_j is the distance between the current input and the j-th input in the sample of training set. The radic σ is again given by

$$\sigma = a n^{-b/n}. \tag{5.2.9}$$

The architecture of the GRNN is similar to PNN, except that the weights in the output layer is not set to 1. Instead they are set to the corresponding values of the output y in the training set. In addition, the sum of the outputs from the Gaussian layer has to be calculated so that the final output can be divided by the sum. The architecture is shown in Figure 5.4 for a single output of two variables input with a set of 10 data points.

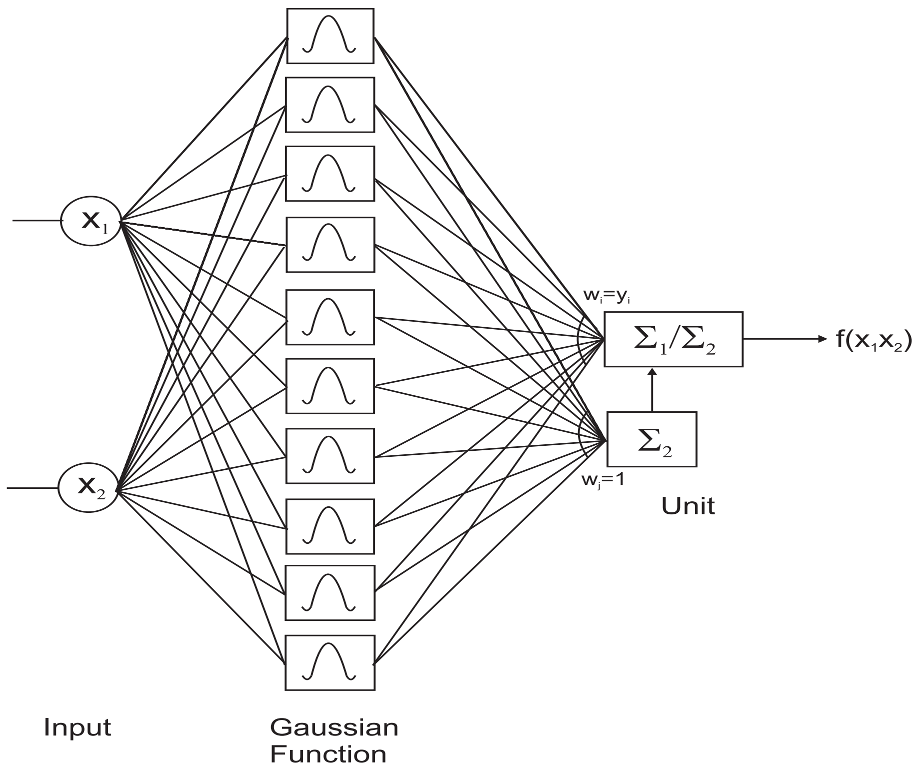


Figure 5.4: General Regression Neural Network Architecture

5.2.3 Generalized Additive Models Networks

The generalized additive model, GAM relating an output y to a vector input $\underset{\sim}{x} = (x_1, \ldots, x_I)$ is

$$E(Y) = \mu \tag{5.2.10}$$

$$h(\mu) = \eta = \sum_{i=0}^{I} f_i(x_i). \tag{5.2.11}$$

This is a generalization of the GLIM model when the parameters β_i are here replaced by functions f_i. Again $h(\cdot)$ is the link function. Nonparametric estimation is used to obtain the unknown functions f_j. They are obtained iteratively by first fitting f_1, then fitting f_2 to the residual $y_i - f_1(x_{1i})$, and so on. Ciampi and Lechevalier (1997) used the model

$$logit(p) = f_1(x_1) + \ldots + f_I(x_I) \tag{5.2.12}$$

as a 2 class classifier. They estimated the functions f using B-splines.

A neural network corresponding to (5.2.12) with two continuous input is shown in Figure 5.5. The first hidden layer consist of two blocks, each corresponding to the transformation of the input with B-splines. A second hidden layer computes the f's functions, all units have as activation the identity function. The output layer has a logistic activation function and output p.

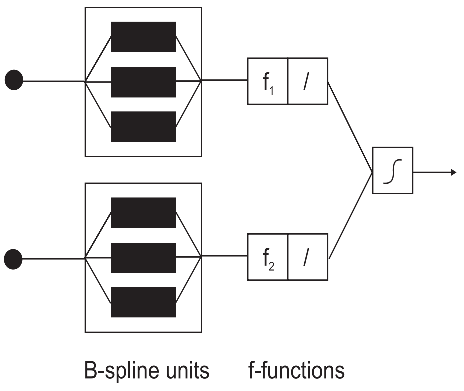
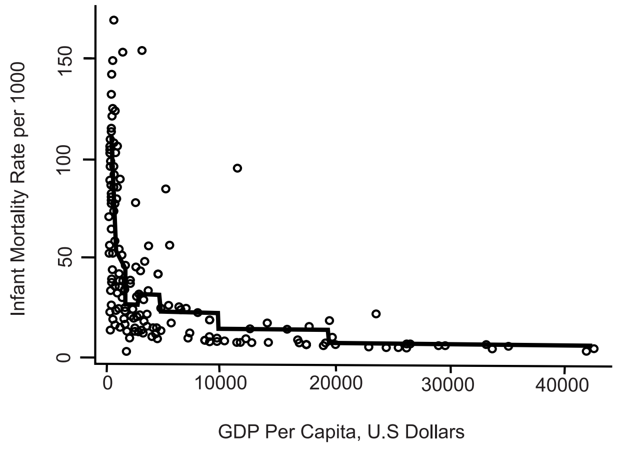


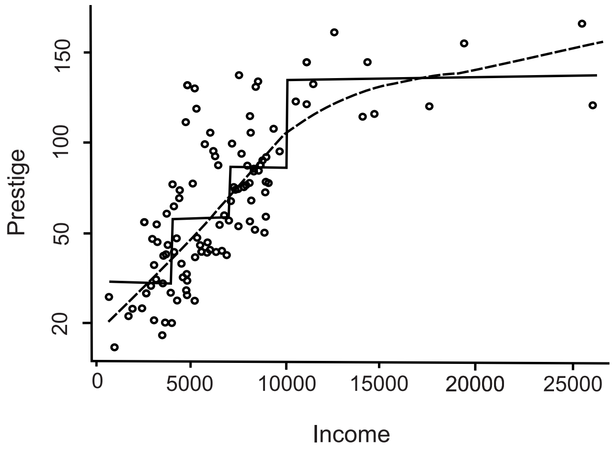
Figure 5.5: GAM neural network

5.2.4 Regression and Classification Tree Network

Regression tree is a binning and averaging procedure. The procedure is presented in Fox (2000a,b) and for a sample regression of y on x, to average the values of y for corresponding values of x for certain conveniently chosen regions. Some examples are shown in Figure 5.6.



(a) The binning estimator applied to the relationship between infant mortality per 1000 and GDP per capita, in US dollars. Ten bins are employed.



(b) The solid line gives the estimated regression function relating occupational prestige to income implied by the tree, the broken line is for a local linear regression.

Figure 5.6: Regression tree $\hat{y}$

Closely related to regression trees are classification trees, where the response variable is categorical rather than quantitative.

The solid line gives the estimated regression function relating occupational prestige to income implied by the tree, the broken line is for a local linear regression.

The regression and classification tree models can be written respectively as:

$$Y = \alpha_1 I_1(\underset{\sim}{x}) + \ldots + I_L(\underset{\sim}{x}) \tag{5.2.13}$$

$$logit(p) = \gamma_1 I_1(\underset{\sim}{x}) + \ldots + I_L(\underset{\sim}{x}) \tag{5.2.14}$$

where the I are characteristics functions of L-subsets of the predictor space which form a partition.

The following classification tree is given in Ciampi and Lechevalier (1997).

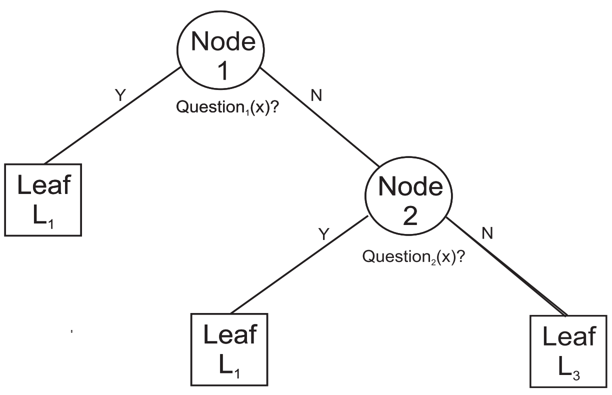


Figure 5.7: Decision Tree

The leaves of the tree represents the sets of the partition. Each leaf is reached through a series of binary questions involving the classifiers (input); these are determined from the data at each mode.

The trees can be represented as neural networks. Figure 5.8 shows such representation for the tree of Figure 5.7.

The hidden layers have activation function taking value -1 for negative input and 1 for positive input. The weights linking the second hidden layer to the output are determined by the data and are given by the logit of the class probability corresponding to the leaves of the tree.

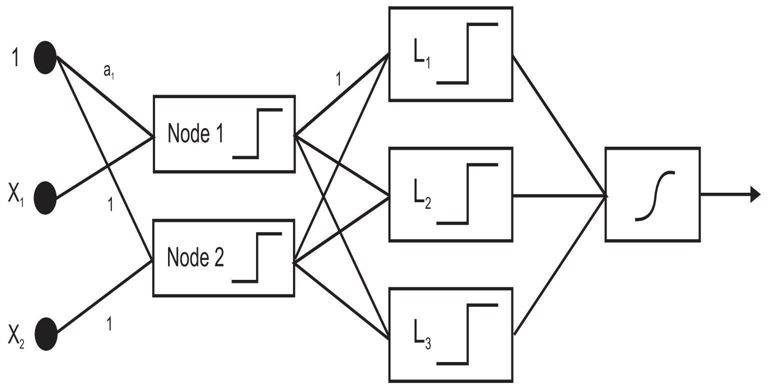


Figure 5.8: Classification Tree Network

5.2.5 Projection Pursuit and Feed-Forward Networks

Generalized additive models fits the model

$$y_i = \alpha + f_1(x_1) + \ldots + f_I(x_I). \tag{5.2.15}$$

Projection pursuit regression (PPR), fits the model

$$y_i = \alpha + f_1(z_{i1}) + \ldots + f_I(z_{iI}) \tag{5.2.16}$$

where the z_1's are linear combinations of the x's

$$z_{ij} = \alpha_{j1}x_{i1} + \alpha_{j2}x_{i2} + \ldots + \alpha_{\alpha p}x_{ip} = \underset{\sim j}{\alpha'}\underset{\sim i}{x}\,. \tag{5.2.17}$$

The projection-pursuit regression model can therefore capture some interactions among the x's. The fitting of this models is obtained by least squares fitting of $\underset{\sim 1}{\hat{\alpha}}$, then as in GAM fitting $\hat{f}_1(\underset{\sim 1}{\hat{\alpha}'}\underset{\sim}{x})$ from the residual R_1 of this model, estimate $\hat{\alpha}_2$ and then $\hat{f}_2(\underset{\sim 2}{\hat{\alpha}}\,\underset{\sim}{x})$ and so on.

The model can be written as

$$y_k = f(\underset{\sim}{x}) = \alpha_k + \sum_{j=1}^{I} f_{ik}(\alpha_j + \underset{\sim j}{\alpha'}\underset{\sim}{x}) \tag{5.2.18}$$

for multiple output $\underset{\sim}{y}$.

Equation (5.2.18) is the equation for a feed-forward network, if, for example, the functions f_j are sigmoid function and we consider the identity output.

Clearly feed-forward neural network is a special case of PPR taking the smooth functions f_j to be logistic functions. Conversely, in PPR we can approximate each smooth function f_i by a sum of logistic functions.

Another heuristic reason why feed-forward network might work well with a few ridden units, for approximation of functions, is that the first stage allows a projection of the $\underset{\sim}{x}$'s onto a space of much lower dimension for the z's.

5.2.6 Example

Ciampi and Chevalier (1997) analyzed a real data set of 189 women who have had a baby. The output variable was 1 if the mothers delivered babies of normal weight and 0 for mothers delivered babies of abnormally low weight (< 2500 g). Several variables were observed in the course of pregnancy. Two continuous (age in years and weight of mother in the last menstrual period before pregnancy) and six binary variables.

They apply a classification tree network, a GAM network, and a combination of the two. Figure 6 gives their classification network. Their GAM and combined network are shown in Figures 7 and 8. The misclassification number resulting in each network was: GAM : 50 - Classification tree : 48 Network of network : 43.

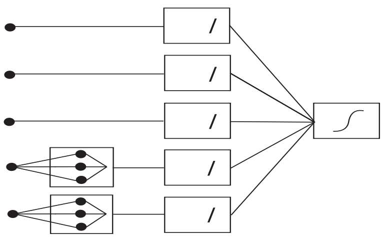


Figure 5.9: GAM network

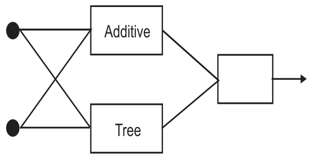


Figure 5.10: Network of networks

Chapter 6

Survival Analysis, Time Series and Control Chart Neural Network Models and Inference

6.1 Survival Analysis Networks

If T is a non-negative random variable representing the time to failure or death of an individual, we may specify the distribution of T by any one of the probability density function $f(t)$, the cumulative distribution function $F(t)$, the survivor function $S(t)$, the hazard function $f(t)$ or the cumulative hazard function $H(t)$. These are related by:

$$\begin{aligned} F(t) &= \int_0^t f(u)du \\ f(t) &= F'(t) = \frac{d}{dt}F(t) \\ S(t) &= 1 - F(t) \\ h(t) &= \frac{f(t)}{S(t)} = -\frac{d}{dt}[\ln S(t)] \\ H(t) &= \int_0^t h(u)du \\ h(t) &= H'(t) \\ S(t) &= \exp[-H(t)]. \end{aligned} \tag{6.1.1}$$

A distinct feature of survival data is the occurrence of incomplete observations. This feature is known as *censoring* which can arise because of time limits and other restrictions depending of the study.

There are different types of censoring.

- Right censoring occur if the events is not observed before the prespecified study-term or some competitive event (e.g. death by other cause) that causes interruption of the follow-up on the individual experimental unit.

- Left censoring happens if the starting point is located before the time of the beginning of the observation for the experimental unit (e.g. time of infection by HIV virus in a study of survival of AIDS patients).

- Interval censoring the exact time to the event is unknown but it is known that it falls in an interval I_i (e.g. when observations are grouped).

The aim is to estimate the previous functions from the observed survival and censoring times. This can be done either by assuming some parametric distribution for T or by using non-parametric methods. Parametric models of survival distributions can be fitted by maximum likelihood techniques. The usual non-parametric estimator for the survival function is the Kaplan-Meier estimate. When two or more group of patients are to be compared the log-rank or the Mantel-Hanszel tests are used.

General class of densities and the non-parametric procedures with estimation procedures are described in Kalbfleish and Prentice (2002).

Usually we have covariates related to the survival time T. The relation can be linear $(\underset{\sim}{\beta}'\underset{\sim}{x})$ or non-linear $(g(\underset{\sim}{j};\underset{\sim}{x}))$. A general class of models relating survival time and covariates is studied in Louzada-Neto (1997, 1999). Here we describe the three most common particular cases of the Louzada-Neto model.

The first class of models is the *accelerated failure time* (AFT) models

$$\log T = -\underset{\sim}{\beta}e'\underset{\sim}{x} + W \tag{6.1.2}$$

where W is a random variable. Then exponentiation gives

$$T = \exp\left(-\beta'\underset{\sim}{x}\right)e^W \text{ or } T' = e^W = T\exp\left(\beta'\underset{\sim}{x}\right) \tag{6.1.3}$$

where T' has hazard function h_0 that does not depend on β. If $h_j(t)$ is the hazard function for the j'th patient it follows that

$$h_j(t) = h_0(t\exp\beta'x)\exp\beta'x. \tag{6.1.4}$$

The second class is the *proportional odds* (PO) where the regression is on the log-odds of survival, correspondence to a linear logistic model with "death" or not by a fixed time t_0' as a binary response.

$$\log\frac{S_j(t)}{1-S_j(t)} = \underset{\sim}{\beta}'\underset{\sim}{x} + \log\frac{S_0(t)}{1-S_0(t)} \tag{6.1.5}$$

or

$$\frac{S_j(t)}{1 - S_j(t)} = \frac{S_0(t)}{1 - S_0(t)} \exp \underset{\sim}{\beta}' \underset{\sim}{x}. \tag{6.1.6}$$

The third class is the "proportional hazard" or Cox regression model (PH).

$$\log h_t(t) = \underset{\sim}{\beta}' \underset{\sim}{x} + \log h_0(t) \tag{6.1.7}$$

$$h_j(t) = h_0(t) \exp \underset{\sim}{\beta}' \underset{\sim}{x}. \tag{6.1.8}$$

Ciampi and Etezadi-Amoli (1985) extended models (6.1.2) and (6.1.7) under a mixed model and not only extend these models but also puts the three models under a more general comprehensive model (Louzado Neto and Pereira 2000). See Figure 6.1.

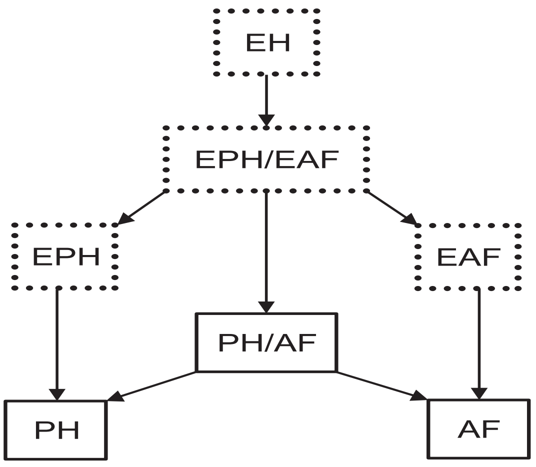


Figure 6.1: Classes of regression model for survival data (.... Louzada-Neto hazard models, __ existing models). PH-proportional hazard, AF-accelerated failure, PH/AM mixed model, EPH-extended PH, EAF-extended AF, EPH/EAF-extended mixed, EH-extended hazard. Each model can be obtained as particular case of models on top of each, in the figure.

Ripley (1998) investigated seven neural networks in modeling breast cancer prognosis; her models were based on alternative implementation of models (6.1.2) to (6.1.5) allowing for censoring. Here we outline the important results of the literature.

The accelerated failure time - AFT model is implemented using the architecture of regression network with the censored times estimated using some missing value method as in Xiang et al (2000).

For the Cox proportional hazard model, Faraggi and Simon (1995) substitute the linear function βx_j by the output $f(x_j, \theta)$ of the neural network, that is

$$L_c(\theta) = \prod_{i\in\ldots} \frac{\exp\left\{\sum_{h=1}^{H} \alpha_h/[1+\exp(-w_h' x_i)]\right\}}{\sum_{j\in R_i} \exp\left\{\sum_{h=1}^{H} \alpha_h/[1+\exp(-w_h' x_i)]\right\}} \tag{6.1.9}$$

and estimations are obtained by maximum likelihood through Newton-Raphson.

The corresponding network is shown below in Figure 6.2.

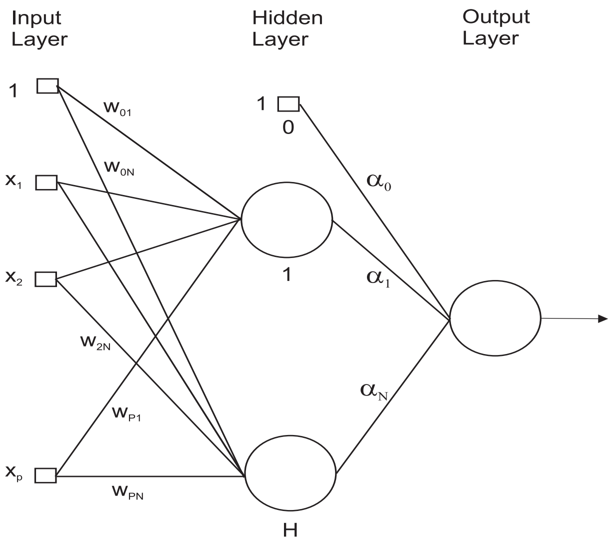


Figure 6.2: Neural network model for survival data (Single hidden layer neural network)

As an example (Faraggi and Simon, 1995) consider the data related to 506 patient with prostatic cancer in stage 3 and 4. The covariates are: stage, age, weight, treatment (0.2; 1 or 5 mg of DES and placebo).

The results are given in the Tables 6.1, 6.2, 6.3 below for the models:

(a) First-order PH model (4 parameters);

(b) Second-order (interactions) PH model (10 parameters);

(c) Neural network model with two hidden nodes (12 parameters);

(d) Neural network model with three hidden nodes (18 parameters).

Table 6.1: Summary statistics for the factors included in the models

	Complete data	Training set	Validation set
Sample size	475	238	237
Stage 3	47.5%	47.6%	47.4%
Stage 4	52.5%	52.4%	52.6%
Median age	73 years	73 years	73 years
Median wight	98-0	97-0	99-0
Treatment: Low	49.9%	48.3%	51.5%
Treatment: High	50.1%	51.7%	48.5%
Median survival	33 months	33 months	34 months
% censoring	28.8%	29.8%	27.7%

Table 6.2: Log-likelihood and c statistics for first-order, second-order and neural network proportional hazards models

Model		Training data		Test data	
	Number of parameter s	Log lik	c	Log lik	c
First order PH	4	-814.3	0.608	-831.0	0.607
Second-order PH	10	-805.6	0.648	-834.8	0.580
Neural network $H = 2$	12	-801.2	0.646	-834.5	0.6000
Neural network $H = 3$	18	-794.9	0.661	-860.0	0.582

Table 6.3: Estimation of the main effects and higher order interactions using 2^4 factorial design contrasts and the predictions obtained from the different models

Effects	PH 1st order	PH 2nd order	Neural network $H = 2$	Neural network $H = 3$
Stage	0.300	0.325	0.451	0.450
Rx*	-0.130	-0.248	-0.198	-0.260
Age	0.323	0.315	0.219	0.278
Weight	-0.249	-0.238	-0.302	-0.581
Stage $\times Rx$	0	-0.256	-0.404	-0.655
State $\times$ Age	0	-0.213	-0.330	-0.415
State $\times Wt$*	0	-0.069	-0.032	-0.109
$Rx\times$ Age	0	0.293	0.513	0.484
$Rx \times Wt$	0	-0.195	-0.025	0.051
Stage $\times Rx\times$ Age	0	0	0.360	0.475
Stage $\times Rx \times Wt$	0	0	0.026	0.345
Stage $\times$ Age $\times Wt$	0	0	-0.024	0.271
$Rx\times$ Age $\times Wt$	0	0	0.006	-0.363
State $\times Rx\times$ Age $\times Wt$	0	0	0.028	-0.128

* Rx = Treatment, Wt = Weight

Implementation of the proportional odds and proportional hazard were implemented also by Liestol et al (1994) and Biganzoli at al (1998).

Liestol, Anderson (1994) used a neural network for Cox's model with covariates in the form.

Let T be a random survival time variable, and I_k the interval $t_{k-1} < t < t_k,\ k = 1, \ldots, K$ where $0 < t_0 < t_1 < \ldots < t_k < \infty$.

The model can be specified by the conditional probabilities.

$$P(T \in I_k/T > t_{k-1}, x) = \frac{1}{1 + \exp\left(-\beta_{0k} - \sum_{i=1}^{1} \beta_{ik}x_i\right)} \tag{6.1.10}$$

for $K = 1, \ldots, K$.

The corresponding neural network is the multinomial logistic network with k outputs.

The output 0_k in the k^{th} output neuron corresponds to the conditional probability of dying in the interval I_k.

Data for the individual n consist of the regressor x^n and the vector $(y_1^n, \ldots, y^n)$ where y_k^n is the indicator of individual n dying in I_k and $k_n \leq K$ is the number of intervals where n is observed. Thus $y_1^n, \ldots, y^n k_{n-1}$ are all 0 and $y^n k_n = 1$ of n dies in I_{kn} and

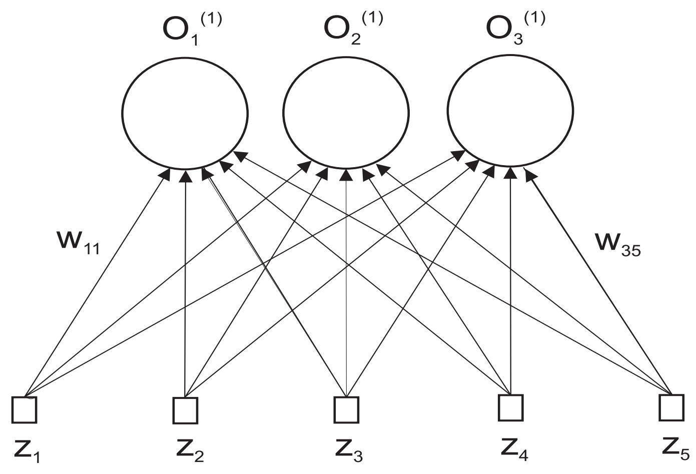


Figure 6.3: Log odds survival network

$$0_k = f(x, w) = \Lambda\left(\beta_{0k} + \sum_{i=1}^{1} \beta_{ik} x_i\right) \tag{6.1.11}$$

and the function to optimize

$$E^*(w) = \sum_{h=1}^{N} \sum_{k=1}^{K_n} - \log\left(1 - |y_k^n - f(x^n, w)|\right) \tag{6.1.12}$$

and $w = (\beta_{01}, \ldots, \beta_{0k}, \beta_{11}, \ldots, \beta_{Ik}$ and under the hypothesis of proportional rates make the restriction $\beta_{1j} = \beta_2 = \beta_{3j} = \beta_{4j} = \ldots = \beta_j$. Other implementations can be seen in Biganzoli et al (1998).

An immediate generalization would be to substitute the linearity for non-linearity of the regressors adding a hidden layer as in the Figure 6.4 below:

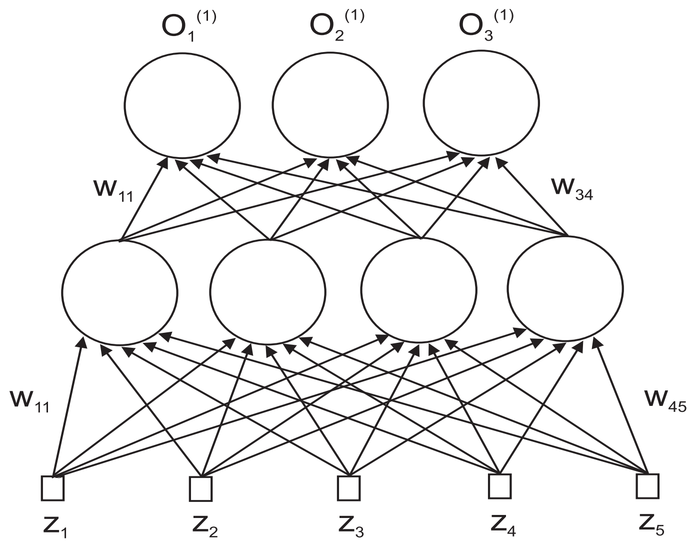


Figure 6.4: Non-linear survival network (Two-layer feed-forward neural nets, showing the notation for nodes and weights: input nodes (•), output (and hidden) nodes (O), covariates Z_j, connection weights w_{ij}, output values $0_1^{(2)}$)

An example from Liestol et al (1994) uses the data from 205 patients with melanoma of which 53 died, 8 covariates were included.

Several networks were studied and the negative of the likelihood (interpreted as prediction error) is given in the Table 6.4 below:

Table 6.4: Cross validation of models for survival with malignant melanoma (Column 1. Linear model; 2. Linear model with weight-decay; 3. Linear model with a penalty term for non-proportional hazards; 4. Non-linear model with proportional hazards; 5. Non-linear model with a penalty term for non-proportional hazards; 6. Non-linear model with proportional hazards in first and second interval and in third and fourth interval; 7. Non-linear model with non-proportional hazards.)

	1	2	3	4	5	6	7
Prediction error	172.6	170.7	168.6	181.3	167.0	168.0	170.2
Change		-1.9	-4.0	8.7	-5.6	-4.6	-2.4

The main results for non-linear models with two hidden nodes were:

- Proportional hazard models produced inferior predictions, decreasing the test log-likelihood of a two hidden node model by 8.7 (column 4) when using the standard weight decay, even more if no weight decay was used.
- A gain in the test log-likelihood was obtained by using moderately non-proportional models. Adding a penalty term to the likelihood of a non-proportional model or assuming proportionality over the two first and last time intervals improved the test log-likelihood by similar amounts (5.6 in the former case (column 5) and 4.6 in the latter (column 6)). Using no restrictions on the weights except weight decay gave slightly inferior results (column 7, improvement 2.4).

In summary, for this small data set the improvements that could be obtained compared to the simple linear models were moderate. Most of the gain could be obtained by adding suitable penalty terms to the likelihood of a linear but non-proportional model.

An example from Biganzoli et al (1998) is the application neural networks in the data sets of Efron's brain and neck cancer and Kalbfleish and Prentice lung cancer using the network architecture of Figure 6.5. The results of the survival curve fits follows in Figures 6.6 and 6.7:

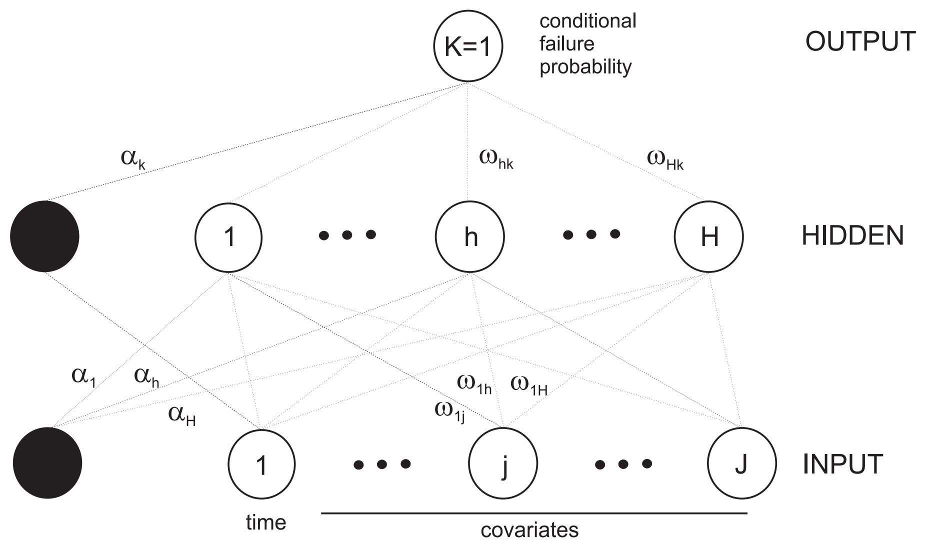
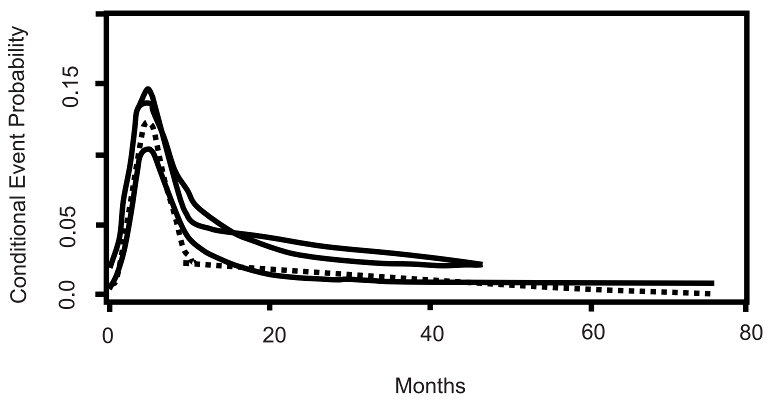
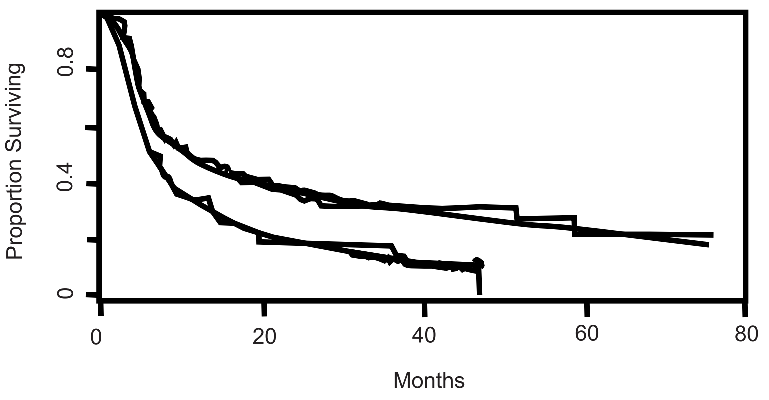


Figure 6.5: Feed forward neural network model for partial logistic regression (PLANN) (The units (nodes) are represented by circles and the connections between units are represented by dashed lines. The input layer has J units for time a and covariates plus one ... unit (0). The hidden layer has H units plus the ... unit (0). A single output unit $(K = 1)$ compute conditional failure probability x_1 and x_2 are the weights for the connections of the ... unit with the hidden and output unit w_a and $w_{...}$ are the weights for the connections between input and hidden units and hidden and output unit, respectively.)

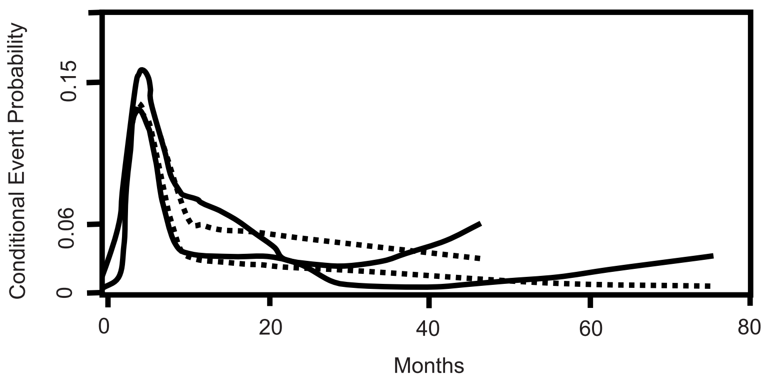


(a)

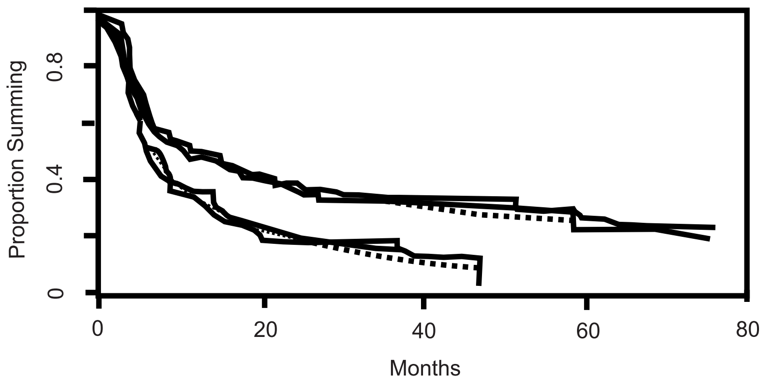


(b)

Figure 6.6: Head and neck cancer trial ((a) estimates of conditional failure probability obtained with the best PLANN configuration (solid line) and the cubic-linear spline proposed by Efron ^{13} (dashed lines); (b) corresponding survival function and Kaplan-Meyer estimates.)



(a)



(b)

Figure 6.7: Head and neck cancer trial ((a) estimates of conditional failure probability obtained with a suboptional PLANN model (solid line) and the cubic-linear spline proposed by Efron [13] (dashed lines); (b) corresponding survival function and Kaplan-Meyer estimates.)

A further reference is Bakker et al (2003) who have used a neural-Bayesian approach to fit Cox survival model using MCMC and an exponential activation function. Other applications can be seen in Lapuerta et al (1995), Ohno-Machodo et al (1995) and Mariani and al (1997).

6.2 Time Series Forecasting Networks

A time series is a set of data observed sequentially in time; although time can be substituted by space, depth etc. Neighbouring observations are dependent and the study of time series data consists in modeling this dependency.

A time series is represented by a set of observations $\{\underset{\sim}{y}(\underset{\sim}{t}), \underset{\sim}{t} \in T\}$ where T is a set of index (time, space, etc.).

The nature of Y and T can be each a discrete or a continuous set. Further, $\underset{\sim}{y}$ can be univariate or multivariate and T can have unidimensional or a multidimensional elements. For example, $(Y, Z)'_{(t,s)}$ can represent the number of cases Y of influenza and Z of meningitis in week t and state s in 2002 in the USA.

Thee are two main objectives in the study of time series. First, to understand the underline mechanism that generates the time series. Second, to forecast future values of it. Problems of interest are:

- description of the behavior in the data,

- to find periodicities in the data,

- to forecast the series,

- to estimate the transfer function $v(t)$ that connects an input series X to the output series Y of the generating system. For example, in the linear case

$$Y(t) = \sum_{u=0}^{\infty} u(t)X(t-u), \tag{6.2.1}$$

- to forecast Y given v and X,

- to study the behavior of the system, by simulating X,

- to control Y by making changes in X.

There are basically two approaches for the analysis of time series. The first, analyzes the series in the time domain, that is we are interested in the magnitude of the events that occur in time t. The main tool used is the autocorrelation function (and functions of it) and we use parametric models. In the second approach, the analysis is in the frequency domain, that is we are interested in the frequency in which the events occur in a period of time. The tool used is the spectrum (which is a transform of the correlation function ex: Walsh, Fourier, Wavelet transforms) and we use non-parametric methods. The two approaches are not alternatives but complementary and are justified by representation theorems due to Wold (1938) in time domain and Cramer (1939) in the frequency domain. We can say from these theorems that the time domain analysis is used when the main interest is in the analysis of *non-deterministic* characteristic of this data and the frequency domain is used when the interest is in the *deterministic* characteristic of the data. Essentially Wold (1938) results in the time domain states that any stationary process Y_t can be represented by

$$Y_t = D_t + Z_t \tag{6.2.2}$$

where D_t and Z_t are uncorrelated, D_t is deterministic that is depends only on its past values $D_{t-1}, D_{t-2}, \ldots$ and Z_t is of the form

$$Z_t = \epsilon_t + \psi_1 \epsilon_{t-1} + \psi_2 \epsilon_{t-2} \tag{6.2.3}$$

or

$$Z_t = \pi_1 Z_{t-1} + \pi_2 Z_{t-2} + \ldots + \epsilon_t \tag{6.2.4}$$

where ϵ_t is a white noise sequence (mean zero and autocorrelations zero) and ψ_1, π_1 are parameters. Expression (6.2.3) is an infinite moving-average process, $MA(\infty)$ and expression (6.2.4) is an infinite autoregressive process $AR(\infty)$.

The frequency domain analysis is based on the pair of results:

$$\rho(t) = \int_{-\infty}^{\infty} e^{iwt} dF(w) \tag{6.2.5}$$

where $\rho(t)$ is the autocorrelation of Y_t and acts as the characteristic function of the (spectral) density $F(w)$ of the process $Z(w)$ and

$$Y(t) = \int_{-\pi}^{\pi} e^{iwt} dZ(w). \tag{6.2.6}$$

Most of the results in time series analysis is based on these results on its extensions.

We can trace in time some developments in time series in the last century: Decomposition Methods (Economics, thirties to fifties), Exponential Smoothing (Operations Research, sixties), Box-Jenkins (Engineering, Statistics, seventies), State-Space Methods - Bayesian, Structural (Statistics, Econometrics, Control Engineering,eighties), Non-Linear Models (Statistics, Control Engineering, nineties).

It should be pointed out that the importance of Box-Jenkins methodology is that they popularized and made accessible the difference equation representation of Wold. Further we can make an analogy between scalar and matrices and the difference equation models of Box-Jenkins (scalar) and state-space models (matrices). The recent interest in non-linear model for time series is that makes neural networks attractive and we turn to this in the next Section.

6.2.1 Forecasting with Neural Networks

The most common used architecture is a feed-forward network with p lagged values of the time series as input and one output. As in the regression model of Section 5.1.2, if there is no hidden layer and the output activation is the identity function the network is equivalent to a linear $AR(p)$ model. If there are one or more hidden layers with sigmoid activation function in the hidden neurons the neural network acts as non-linear autoregressions.

The architecture and implementation of most time series applications of neural networks follows closely to the regression implementation. In the regression neural network we have the covariates as inputs to the feed-forward network, here we have the lagged values of the time series as input.

The use of artificial neural network applications in forecasting is surveyed in Hill et al (1994) and Zang et al (1998). Here we will describe some recent and non-standard applications.

One of the series analyzed by Raposo (1992) was the famous Airline Passenger Data of Box-Jenkins (Monthly number of international passengers from 1949 to 1960). The model for this series becomes the standard benchmark model for seasonal data with trend: the multiplicative seasonal integrated moving average model, SARIMA $(0, 1, 1)_{12}(0, 1, 1)$.

Table 6.5 gives the comparison for the periods 1958, 1959, 1960 in terms of mean average percentage error of the Box-Jenkins model and some neural network.

The airline data was also analyzed by Faraway and Chatfield (1998) using neural network.

Another classical time series the Yearly Sunspot Data was analyzed by Park et al (1996). Their results are presented in Tables 6.6 and 6.7 for the training data (88 observations) and forecasting period (12 observations). Here they used the PCA method described in Section 2.8.

A multivariate time series forecasting using neural network is presented in Chakraborthy et al (1992). The data used is the Indices of Monthly Flour Prices in Buffalo (x_t), Minneapolis (y_t) and Kansas City (z_t) over the period from August

Table 6.5: Forecasting for Airline Data

Year	MAPE %			
	(12:2:1)	(13:2:1)	14:2:1)	SARIMA
1958	7.63	6.44	5.00	7.75
1959	6.32	4.90	7.75	10.86
1960	7.39	7.93	11.91	15.55

(a:b:c) - number of neurons in: a - input, b - hidden, c - output, layer

Table 6.6: Result from network structures and the AR(2) model

Network Structure	Mean Square Error (MSE)	Mean Absolute Error[7] (MAE)
2:11:1	156.2792	9.8539
2:3:1	154.4879	9.4056
2:2:1*	151.9368	9.2865
2:1:1	230.2508	12.2454
2:0:1¨	233.8252	12.1660
AR(2)	222.9554	11.8726

* indicates the proposed optimal structure.
¨ represents a network without hidden units.

Table 6.7: Summary of forecasting errors produced by neural network models and the AR(2) model based on untrained data

Model	Mean Square Error (MSE)	Mean Absolute Error (MAE)
2:11:1	160.8409	9.2631
2:3:1	158.9064	9.1710
2:2:1*	157.7880	9.1522
2:1:1	180.7608	11.0378
2:0:1	180.4559	12.1861
AR(2)	177.2645	11.5624

* represents the selected model by the PCA method.

1972 to November 1980, previously analyzed by Tiag and Tsay using an ARMA (1,1) multivariate model. They used two classes of feed-forward networks: separate modeling of each series and a combined modeling as in Figure 6.8 and Figure 6.9.

The results are shown in Tables 6.8 and 6.9.

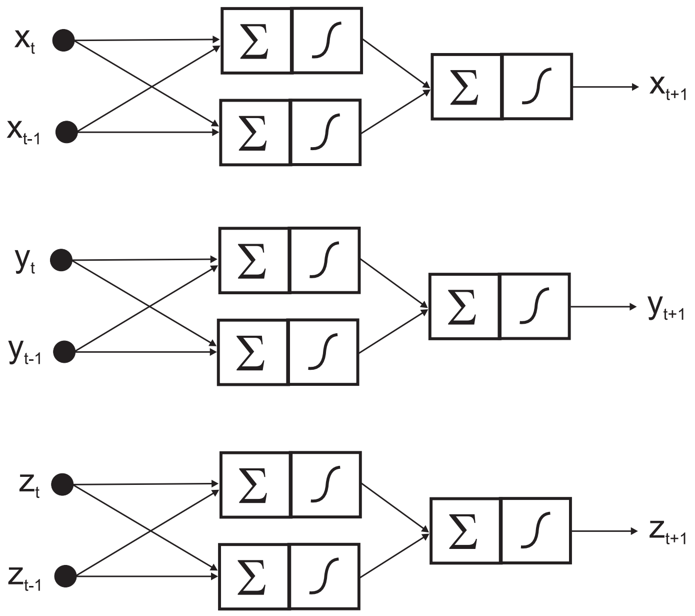


Figure 6.8: Separate Architecture

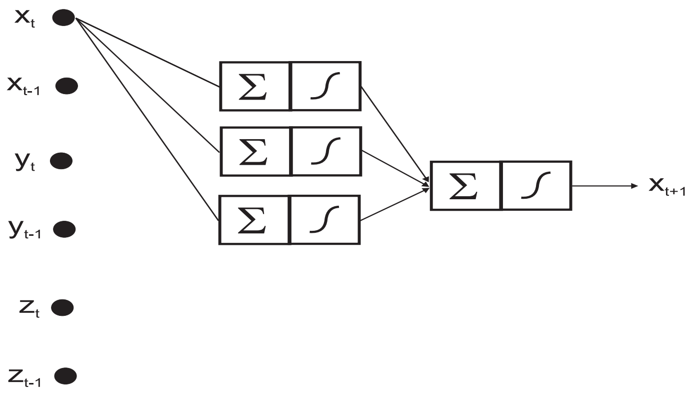


Figure 6.9: Combined Architecture for forecasting x_{t+1} (and similar to y_{t+1} and z_{t+1})

Table 6.8: Mean-Squared Errors for Separate Modeling & Prediction $\times 10^3$

		Buffalo	Minneapolis	Kansas City
Network		MSE	MSE	MSE
2-2-1	Training	3.398	3.174	3.526
	One-lag	4.441	4.169	4.318
	Multi-lag	4.483	5.003	5.909
4-4-1	Training	3.352	3.076	3.383
	One-lag	9.787	8.047	3.370
	Multi-lag	10.317	9.069	6.483
6-6-1	Training	2.774	2.835	1.633
	One-lag	12.102	9.655	8.872
	Multi-lag	17.646	14.909	13.776

Table 6.9: Mean-Squared Errors and Coeffs.of Variation for Combined vs. Tiao/Tsay's Modeling/ Prediction $\times 10^3$

		Buffalo		Minneapolis		Kansas City	
Model		MSE	CV	MSE	CV	MSE	CV
6-6-1	Training	1.446	7.573	1.554	7.889	1.799	8.437
Network	One-lag	3.101	11.091	3.169	11.265	2.067	9.044
	Multi-lag	3.770	12.229	3.244	11.389	2.975	10.850
8-8-1	Training	0.103	2.021	0.090	1.898	0.383	3.892
Network	One-lag	0.087	1.857	0.072	1.697	1.353	7.316
	Multi-lag	0.107	2.059	0.070	1.674	1.521	7.757
Tiao/Tsay	Training	2.549	10.054	5.097	14.285	8.645	18.493
	One-lag	2.373	9.701	4.168	12.917	7.497	17.222
	Multi-lag	72.346	53.564	137.534	74.204	233.413	96.096

In previous example, initial analysis using true series methods have helped to design the architecture of the neural network. More explicit combination of time series model and neural network are: Tseng et al (2002) that improved the ARIMA forecasting using a feed-forward network with logged values of the series, its forecast and the residual obtained from the ARIMA fitting to the series; Coloba et al (2002) which used a neural network after decomposing the series from its law and high frequency signals; Cigizoglu (2003) used generated data from a time series model to train a neural network. Periodicity terms and lagged values were used to forecast river flow in Turkey.

A comparison of structural time series model and neural network is presented in Portugal (1995). Forecasting non-inversible time series using neural network was attempted by Pino et al (2002) and combining forecasts by Donaldson and Kamstra (1996). Further comparisons with time series models and applications to classical time series data can be seen in Terasvirta et al (2005), Kajitoni et al (2005) and Ghiassi et al (2005).

Modeling of non-linear time series using neural network and an extension called stochastic neural network can be seen in Poli and Jones (1994), Stern (1996), Lai and Wong (2001) and Shamseldin and O'Connor (2001).

Forecasting using a combination of neural networks and fuzzy set are presented by Machado et al (1992), Lourenco (1998), Li et al (1999).

Finally, an attempt to choose a classical forecast method using a neural network is shown in Venkatachalan and Sohl (1999).

6.3 Control Chart Networks

Control charts are used to identify variations in production processes. There are usually two types of causes of variation: chance causes and assignable causes. Chance causes, or natural causes, are inherent in all processes. Assignable causes represent improperly adjustable machines, operation error, and defective raw materials and it is desirable to remove these causes from the process.

The process means (μ) and variance (σ^2) are estimated when observed data are collected in sample of size n and the means $\bar{X}_1, \bar{X}_2, \ldots$ are calculated. It is assumed that the sample means are mutually independent and that $\bar{X}_i$ has distribution $N(\mu_j, \frac{\sigma^2}{n})$, $i = 1, 2, \ldots$ and the condition $\mu = \mu_0$ is to be maintained. A shift in the process mean occurs when the distribution of the sample changes $(\mu \neq \mu_0)$. The decision in the control chart is based on the sample mean with a signal of out-of-control being given at the first stage N such that

$$\delta_N = \frac{|\bar{X}_N - \mu_0|}{\sigma/\sqrt{n_0}} > c \tag{6.3.1}$$

where usually $c = 3$ and the sample size of 4 or 5. This procedure is useful to detect shifts in the process mean. However, disturbances can also cause shifts or changes in the amount of process variability. A similar decision procedure can be used for the behavior of ranges of the samples from the process and will signal an out-of-control variability. Similarly chart for σ is usually based on

$$|\bar{R} - \sigma_R| > 3 \tag{6.3.2}$$

or

$$\bar{R} \pm 3A_m\bar{R} \tag{6.3.3}$$

where $\bar{R}$ is average range of a number of sample ranges R and A_n is a constant which depends on n, relating σ to $E(R)$ and $V(R)$.

Several kinds of signals based in these statistics and charts have been proposed, for example: more than threes consecutive values more than 2σ from the mean; eight consecutive values above the mean etc. This rules will detect trends, sinoidal etc. patters in the data.

These rules can also be implemented using neural network and we will now describe some of these work. All the them uses simulation results and make comparison with classical statistical charts in terms of percentage of wrong decisions and ARL (average run length) which is defined to be the expected number of samples that are observed from a process before out-of-control signal is signaled by the chart.

Prybutok et al (1994) used a feed-forward network with 5 inputs (the sample size), a single hidden layer with 5 neurons and 1 output neuron. The activation functions were sigmoid the neural network classifier the control chart as being "in control" or "out-of-control" respectively according if the output fall in the intervals (0, 0.5) or (0.5, 1).

Various rules were compared (see Prybutok et al (1994)) such as C_4=signal if eight if the last standardized sample means are between 0 and -3 or if the eight of the last eight are between 0 and 3. The rules were denoted by C_i and its combination by C_{ijkl}.

For eight shifts given by $|\mu - \mu_0|$ taking the values (0, 0.2, 0.4, 0.6, 0.8, 1, 2, 3) the ratio of the ARL for the neural network to the standard Shewhart $\bar{X}$ control chart without supplemental runs rules are listed in Table 6.10. A further table with other rules are given in Prybutok et al (1994).

Table 6.10: Rations of Average Run Lengths: Neural Network Model Cn_{ij} to Shewhart Control Chart without Runs Rules (C_1)

Control	Shift (μ)							
Charts	0.0	0.2	0.4	0.6	0.8	1.0	2.0	3.0
ARL^*	370.40	308.43	200.08	119.67	71.55	43.89	6.30	2.00
C_1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Cn_{54}	0.13	0.14	0.16	0.18	0.21	0.24	0.27	0.19
Cn_{55}	0.25	0.27	0.30	0.33	0.38	0.38	0.46	0.32
Cn_{56}	0.47	0.55	0.60	0.67	0.68	0.72	0.71	0.44
Cn_{57}	0.95	0.99	1.31	1.41	1.48	1.44	1.19	0.68
Cn_{62}	0.18	0.19	0.22	0.25	0.30	0.32	0.43	0.34
Cn_{63}	0.24	0.26	0.29	0.32	0.37	0.43	0.50	0.36
Cn_{64}	0.33	0.34	0.36	0.39	0.44	0.48	0.55	0.43
Cn_{65}	0.42	0.40	0.44	0.48	0.53	0.56	0.67	0.53
Cn_{66}	0.56	0.53	0.57	0.60	0.63	0.69	0.76	0.56
Cn_{67}	0.78	0.71	0.72	0.78	0.82	0.79	0.90	0.63
Cn_{68}	1.15	1.02	0.94	1.01	1.02	1.08	1.04	0.75
Cn_{71}	0.25	0.26	0.27	0.33	0.33	0.38	0.44	0.07
Cn_{72}	0.49	0.52	0.52	0.60	0.61	0.63	0.70	0.42
Cn_{73}	0.88	0.92	0.96	1.01	1.02	1.09	0.98	0.63
Cn_{74}	1.56	1.61	1.71	1.72	1.69	1.56	1.47	0.90

$ARL = ARL_{c_1}(\mu)$ for Shewhart Control Chart.

Smith (1994) extends the previous work by considering a single model of a simultaneous X-bar and R chart, and investigating both location and variance shifts. She applied two groups of feed-forward networks.

The first group with only one output, sigmoid activation function, two hidden layers, 3 or 10 neurons in each hidden layer, and interpretation as: 0 to 0.3 (mean shift), 0.3 to 0.7 (under control), 0.7 to 1 (variance shift).

The inputs were either in ten observations $(n = 10)$ and the calculated statistics (sample mean, range, standard deviation), a total of 13 inputs or on the calculated islatistics only, i.e. 3 inputs. Table 6.11 shows the percentage of wrong decisions for the control chart and the neural networks.

Table 6.11: Wrong Decisions of Control Charts and Neural Networks

Training/ Test Set	Inputs	Hidden Layer Size	Neural Net Percent Wrong	Shewhart Control Percent Wrong
A	All	3	16.8	27.5
A	All	10	11.7	27.5
A	Statistics	3	28.5	27.5
B	All	3	0.8	0.5
B	All	10	0.5	0.5
B	Statistics	3	0.0(None)	0.5

Table 6.12: Desired Output Vectors

Pattern	Desired output vector	Note
Upward trend	[1,0,0,0,0]	–
Downward trend	[-1,0,0,0,0]	–
Systematic variation (I)	[0,1,0,0,0]	the first observation is above the in-control mean
Systematic variation (II)	[0,-1,0,0,0]	the first observation is below the in-control mean
Cycle (I)	[0,0,1,0,0]	sine wave
Cycle (II)	[0,0,-1,0,0]	cosine wave
Mixture (I)	[0,0,0,1,0]	the mean of the first distribution is greater than the in-control mean
Mixture (II)	[0,0,0,-1,0]	the mean of the first distribution is less than the in-control mean
Upward shift	[0,0,0,0,1]	–
Downward shift	[0,0,0,0,-1]	–

The second group of network deferred from the first group by the input which now is the raw observations only $(n = 5, n = 10, n = 20)$. These neural networks were design to recognize the following shapes: flat (in control), sine wave (cyclic), slop (trend of drift) and bimodal (up and down shifts). The networks here had two outputs used to classify patterns with the codes: flat (0,0), sine wave (1,1), trend (0,1), and bimodal (1,0). A output in (0,0.4] was considered 0 and in (0.6,1) was considered 1. Table 6.3 gives some of the results Smith (1994) obtained.

Hwarng (2002) presents a neural network methodology for monitoring process shift in the presence of autocorrelation. The study of AR (1) (autoregressive of order one) processes shows that the performance of neural network based monitoring scheme is superior to other five control charts in the cases investigated. He used a feed-forward network with $50 \cdot 30 \cdot 1$ (input·hidden·output) structure of neurons.

A more sophisticated pattern recognition process control chart using neural networks is given in Cheng (1997). A comparison between the performance of a feed-forward network and a mixture of expert was made using simulations. The multilayer network had 16 inputs (raw observations), 12 neurons in the hidden layers and 5 outputs identifying the patterns as given in Table 6.13. Hyperbolic activation function was used.

A mixture of three expert (choice based on experimentation) with the same input as in the feed-forward network. Each expert had the same structure as the feed-forward network. The gating network has 3 neurons elements in the output layer and 4 neurons in a hidden layer.

The results seems to indicate that the mixture of experts performs better in terms of correctly identified patterns.

Table 6.13: Pattern Recognition Networks

# Input	σ	# Correct	# Missed of 300			
Points	of Noise	of 300	Flat	Slope	Sine wave	Bimodal
5	0.1	288	0	0	10	2
5	0.2	226	39	15	10	10
5	0.3	208	32	39	18	3
10	0.1	292	0	0	5	3
10	0.2	268	3	7	16	6
10	0.3	246	9	17	23	5
20	0.1	297	0	0	0	3
20	0.2	293	0	0	0	7
20	0.3	280	6	1	1	12
Over All Networks		88.8%	3.3%	2.9%	3.1%	1.9%

6.4 Some Statistical Inference Results

In this section, we describe some inferential procedures that have been developed for and using neural network. In particular we outline some results on point estimation (Bayes, likelihood, robust), interval estimation (parametric, bootstrap) and significance tests (F-test and nonlinearity test).

A more detailed account of prediction interval outlined here see also Hwang and Ding (1997).

6.4.1 Estimation methods

The standard method of fitting a neural network is *backpropagation* which is a gradient descent algorithm to minimize the soma of squares. Here we outline some details of other alternative methods of estimation.

Since the most used activation function is the sigmoid we present some further results. First consider, maximum likelihood estimation, here we follow Schumacher et al (1996) and Faraggi and Simon (1995).

The logistic (sigmoid) activation relates output y to input x assuming

$$P_{\underset{\sim}{x}}(Y = 1/\underset{\sim}{x}) = \wedge\left(\omega_0 + \sum_{i=1}^{I} \omega_i x_{ji}\right) = p(\underset{\sim}{x}, \omega) \tag{6.4.1}$$

where $\wedge(u) = 1/(1 + e^u)$ is the sigmoid or logistic function.

As seen in Section 5.1, this activation function as a non-linear regression model is a special case of the generalized linear model i.e.

$$E(Y/x) = p(x, \omega),\ V(Y/\underset{\sim}{x}) = p(x, \omega)(1 - p(x, \omega)). \tag{6.4.2}$$

The maximum likelihood estimator of the weights ω's is obtained from the log likelihood function

$$L(\omega) = \sum_{j=1}^{n} \Big[y_j \log p(x_i, \underset{\sim}{\omega}) + (1 - y_j)\log(1 - p(x_j\omega)\Big] \tag{6.4.3}$$

where (x_j, y_j) is the observation for pattern j.

To maximize $L(\omega)$ is equivalent to minimize the Kulback-Leiber divergence

$$\sum_{j=1}^{n} \left[y_j \log \frac{y_j}{p(x_j\omega)} + (1 - y_j)\log \frac{1 - y_j}{1 - p(x_j, \omega)}\right] \tag{6.4.4}$$

It can be written as

$$\sum_{j=1}^{n} - \log \left(1 - |y_j - p(x, \omega)|\right) \tag{6.4.5}$$

which makes the interpretation of distance between y and $p(y, \omega)$ more clear. At this point, for comparison, we recall the criteria function used in the backpropagation least square estimation method (or learning rule):

$$\sum_{j=1}^{n} (y_j - p(x_j, \omega))^2 \tag{6.4.6}$$

Due to the relation to of maximum likelihood criteria and the Kulback-Leiber, the estimation based on (6.4.4) is called backpropagation of maximum likelihood. In the neural network jargon it is also called relative entropy or cross entropy method.

Bayesian methods for neural networks rely mostly in MCMC (Markov Chain Monte Carlo). Here we follow Lee (1998) and review the priors used for neural networks.

All approaches start with the same basic model for the output y

$$y_i = \beta_0 + \sum_{j=1}^{k} \beta_j \frac{1}{1 + \exp(-\omega_{j0} - \sum_{h=1}^{p} \omega_{jh} x_{ih})} + \epsilon_i, \tag{6.4.7}$$

$$\epsilon_i \sim N(0, \sigma^2) \tag{6.4.8}$$

The Bayesian models are:

– **Müller and Rios Insua** (1998)
  A direct acyclic graph (DAG) diagram of the model is:

  The distributions for parameters and hyperparameters are:

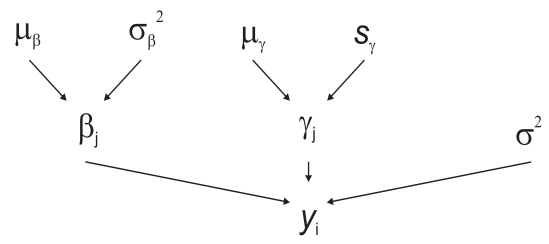


Figure 6.10: Graphical model of Muller and Rios Issua

$$y_i \sim N\left(\sum_{j=0}^{m} \beta_j \wedge (\omega_j' x_i), \sigma^2\right), \; i = 1, \dots, N \tag{6.4.9}$$

$$\beta_j \sim N(\mu_\beta, \sigma_\beta^2), \; j = 0, \dots, m \tag{6.4.10}$$

$$\omega_j - \omega\gamma_j \sim N_p(\mu_\gamma, \mathbf{S}_\gamma), \; j = 1, \dots, m \tag{6.4.11}$$

$$\mu_\beta \sim N(a_\beta, A_\beta) \tag{6.4.12}$$

$$\mu_\gamma \sim N_p(a_\gamma, \mathbf{A}_\gamma) \tag{6.4.13}$$

$$\sigma_\beta^2 \sim \Gamma^{-1}\left(\frac{c_\beta}{2}, \frac{C_\beta}{2}\right) \tag{6.4.14}$$

$$\mathbf{S}_\gamma \sim Wish^{-1}\left(c_\gamma, (c_\gamma \mathbf{C}_\gamma)^{-1}\right) \tag{6.4.15}$$

$$\sigma^2 \sim \Gamma^{-1}\left(\frac{s}{2}, \frac{S}{2}\right). \tag{6.4.16}$$

There are fixed hyperparameters that need to be specified. Müller and Rios Insua specify many of them to be of the same scale as the data. As some fitting algorithms work better when (or sometimes only when) the data have been re-scaled so that $|x_{ih}|, |y_i| \leq 1$, one choice of starting hyperparameters is: $a_\beta = 0$, $A_\beta = 1$, $a_\gamma = \mathbf{0}$, $\mathbf{A}_\gamma = \mathbf{I}_p$, $c_\beta = 1$, $C_\beta = 1$, $c_\gamma = p + 1$, $\mathbf{C}_\gamma = \dfrac{1}{p+1}\mathbf{I}_p, \; s = 1, \; S = \dfrac{1}{10}$.

– **Neal Model** (1996)

  Neal's four-stage model contains more parameters than the previous model. A DAG diagram of the model is

  Each of the original parameters (β and γ) is treated as a univariate normal with mean zero and its own standard deviation. These standard deviations

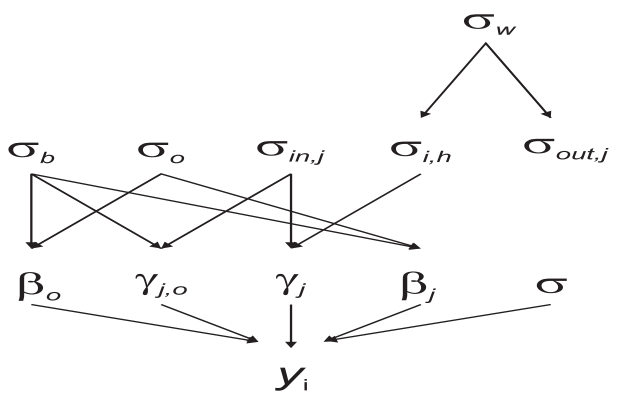


Figure 6.11: Graphical model of Neal

are the product of two hyperparameters, one for the originating node of the link in the graph, and one for the destination node.

There are two notes on this model that should be mentioned. First, Neal uses hyperbolic tangent activation functions rather than logistic functions. These are essentially equivalent in terms of the neural network, the main difference being that their range is from -1 to 1 rather than 0 to 1. The second note is that Neal also discusses using t distributions instead of normal distributions for the parameters.

– **MacKay Model** (1992)
MacKay starts with the idea of using an improper prior, but knowing that the posterior would be improper, he uses the data to fix the hyperparameters at a single value. Under his model, the distributions for parameters and hyperparameters are:

$$y_i \sim N\left(\sum_{j=0}^{m} \beta_j \wedge (\omega_j' x_i)(\gamma_j' x_i), \frac{1}{\nu}\right), \ i = 1, \ldots, N \tag{6.4.17}$$

$$\beta_j \sim N\left(0, \frac{1}{\alpha_1}\right), \ j = 0, \ldots, m \tag{6.4.18}$$

$$\omega_j h \sim N_p\left(0, \frac{1}{\alpha_2}\right), \ j = 1, \ldots, m \ h = 1, \ldots, p \tag{6.4.19}$$

$$\omega_j 0 \sim N_p\left(0, \frac{1}{\alpha_3}\right), \ j = 1, \ldots, m \tag{6.4.20}$$

$$\alpha_k \propto 1, \ k = 1, 2, 3 \tag{6.4.21}$$

$$v \propto 1. \tag{6.4.22}$$

MacKay uses the data to find the posterior mode for the hyperparametrics α and ν and then fixes them at their modes.

Each of the three models reviewed is fairly complicated and are fit using MCMC.

– **Lee Model** (1998)
The model for the response y is

$$y_i = \beta_0 + \sum_{j=1}^{k} \beta_j \frac{1}{1 + \exp(-\gamma_{j0} - \sum_{h=1}^{p} \omega_{jh} x_{ih})} + \epsilon_i, \qquad \epsilon_i \overset{iid}{\sim} N(0, \sigma^2) \tag{6.4.23}$$

Instead of having to specify a proper prior, he uses the noninformative improper prior

$$\pi(\beta, \omega, \sigma^2) = (2\pi)^{-d/2} \frac{1}{\sigma^2}, \tag{6.4.24}$$

where d is the dimension of the model (number of parameters). This prior is proportional to the standard noninformative prior for linear regression. This model is also fitted using MCMC.

This subsection ends with some references on estimation problems of interest. Belue and Bower (1999) give some experimental design procedures to achieve higher accuracy on the estimation of the neural network parameters. Copobianco (2000) outlines robust estimation framework for neural networks. Aitkin and Foxall (2003) reformulate the feedforward neural network as a latent variable model and construct the likelihood which is maximized using finite mixture and the EM algorithm.

6.4.2 Interval estimation

Interval estimation for feedforward neural network essentially uses the standard results of least squares for nonlinear regression models and ridge regression (regularization). Here we summarize the results of Chryssoloures et al (1996) and De Veux et al (1998).

Essentially they used the standard results for nonlinear regression least square

$$S(\omega) = \sum (y_i - f(X,\omega))^2 \tag{6.4.25}$$

or the regularization, when weight decay is used

$$S(\omega) = \sum (y_i - f(X,\omega))^2 + k\alpha \sum \omega_i^2. \tag{6.4.26}$$

The corresponding prediction intervals are obtained from the t distribution and

$$t_{n-p} s\sqrt{1 + g_0'(JJ)^{-1} g_0} \tag{6.4.27}$$

$$t_{n-p^*} s^* \sqrt{1 + g_0'(J'J + kI)^{-1}(J'J + kI)^{-1} g_0} \tag{6.4.28}$$

where $s =$ (residual sum of squares)$/(n-p) = RSS/(n-p)$, p is the number of parameters, J is a matrix with entries $\partial f(\omega, x_i)/\partial \omega_j$ g_0 is a vector with entries $\partial f(x_0 \omega)/\partial \omega_j$ and

$$k = y - f(X\omega^*) + J\omega^*, \tag{6.4.29}$$

$$s^* = RSS/(n - tr(2H - H^2)) \tag{6.4.30}$$

$$H = J(J'J + \alpha I)^{-1} J \tag{6.4.31}$$

$$p^* = tr(2H - H^2). \tag{6.4.32}$$

where $H = J(J'J + \alpha I)^{-1} J'$, ω^* is the true parameter value.

The author gives an application and De Veux et al (1998) also shown some simulated results.

Tibshirani (1995) compared sevens methods to estimate the prediction intervals for neural networks. The method compared were: 1. Delta-method; 2. Approximate delta: Using approximate to information matrix that ignores second derivatives; 3. The regularization estimator; 4. Two approximate sandwich estimators and 5. Two bootstrap estimators.

In this 5 examples and simulations he found that the bootstrap methods provided the most accurate estimates of the standard errors of the predicted values. The non-simulation methods (delta, regularization, sandwich estimators) missed the variability due to random choice of starting values.

6.4.3 Statistical Tests

Two statistical tests when using neural networks are presented here.

Adams (1999) gives the following test procedure to test hypothesis about the parameters (weights) of a neural network. The test can be used to test nonlinear relations between the inputs and outputs variables. To implement the test it is first necessary to calculate the degree of freedom for the network.

If n is the number of observations and p the number of estimated parameters then the degree of freedom (df) are calculated as

$$df = n - p. \tag{6.4.33}$$

With a multilayer feedforward network with m hidden layers, the input layer designed by 0 and the out designed by $m + 1$. For L_{m+1} outputs and L_m neurons in the hidden layer we have

$$df = n - p + (L_{m+1} - 1)(L_n + 1). \tag{6.4.34}$$

The number of parameters is

$$P = \sum_{i=1}^{m+1} (L_{i-1}L_1 + L_i) . \tag{6.4.35}$$

For a network with 4:6:2, i.e. 4 inputs, 6 hidden neurons and 2 outputs, the degrees of freedom for each output is

$$df = n - 44 + (2 - 1)(6 + 1) = n - 37. \tag{6.4.36}$$

For a feedforward network trained by least square we can form the model sum of squares. Therefore any submodel can be tested using the F test

$$F = \frac{(SSE_0 - SSE)/(df_0 - df)}{SSE/df} \tag{6.4.37}$$

where SSE_0 is the sum of squares of submodel and SSE is the sum of squares of full model.

The above test is based on the likelihood ratio test. An alternative proposed by Lee et al (1993) uses the Rao score test or Lagrange multiplier test. Blake and Kapetanios (2003) also suggested a Wald equivalent test. See also Lee (2001).

The implementation of Rao test (see Kiani, 2003) to test nonlinearity of a time series using a one-hidden layer neural network is

(i) regress y_t on intercept and $y_{t-1}, \dots, y_{t-k}$, save residuals $(\hat{u}_t)$, residual sum of squares SSE_1 and predictions $\hat{y}_t$,

(ii) regress $\hat{u}_t$ on $y_{t-1}, \dots, y_{t-k}$ using neural network model that nests a linear regression $y_t = \pi \underset{\sim t}{y} + u_t(\underset{\sim t}{y} = (y_{t-1}, \dots, y_{t-k})$, save matrices Ψ of outputs of the neurons for principle components analysis.

(iii) regress $\hat{e}_t$ (residuals of the linear regression in (ii)) in Ψ^* (principal components of Ψ) and X matrices, and obtain R^2, thus

$$TS = nR^2 \sim \chi^2(g) \tag{6.4.38}$$

where g is the number of principal components used un step (iii).

The neural network in 6.12 clarifies the procedure:

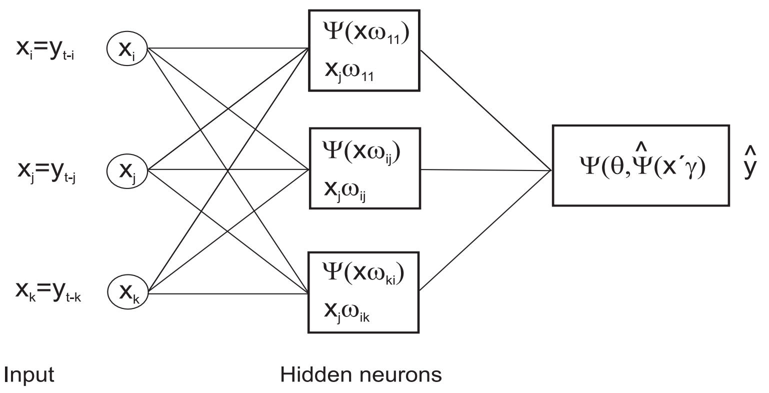


Figure 6.12: One-hidden layer feedforward neural network.

Simulations results on the behaviors of this test can be seen in the bibliography on references. An application is given in Kiani (2003).

Bibliography and references

Preface

Fundamental Concepts on Neural Networks

Artificial Intelligence: Symbolist & Connectionist

[1] Carvalho, L.V. *Data mining*. Editora Erika, SP. Brasil, 1997.

[2] Gueney, K. *An Introduction to Neural Networks*. University College London Press, 1997.

[3] Wilde, D. *Neural Network Models*. Springer, 2nd ed., 1997.

Artificial Neural Networks and Diagrams

[1] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd ed., 1999.

[2] Sarle, W.S. Neural networks and statistical models. In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. 1994. (http//citeseer.nj.nec.com/sarle94neural.html).

Activation Functions

[1] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd ed., 1999.

[2] Igengar, S.S., Chao, E.C., and Phola, V.V. *Foundations of Wavelet Networks and Applications*. Chapman Hall/CRC, 2nd ed., 2002.

[3] Orr, M.J.L. Introduction to radial basis function networks. Tech. rep., Institute of Adaptive and Neural Computationy, Edinburgh University, 1996. (www.anc.ed.ac.uk/ mjo/papers/intro.ps).

Network Architectures

[1] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd ed., 1999.

Network Training

[1] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd ed., 1999.

Kolmogorov Theorem

[1] Bose, N.K. and Liang, P. *Neural Networks Fundamentals with Graphs, Algorithms and Applications*. McGraw-Hill Inc., 2nd ed., 1996.

[2] Mitchell, T.M. *Machine Learning*. McGraw-Hill Inc., 2nd ed., 1997.

[3] Murtagh, F. Neural networks and related “massively parallel” methods for statistics: A short overview. *International Statistical Review*, 62(3):275–288, 1994.

[4] Rojas, R. *Neural Networks: A Systematic Introduction*. Springer, 2nd ed., 1996.

Model Choice

[1] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd ed., 1999.

[2] Murtagh, F. Neural networks and related “massively parallel” methods for statistics: A short overview. *International Statistical Review*, 62(3):275–288, october 1994.

[3] Orr, M.J.L. Introduction to radial basis function networks. Tech. rep., Institute of Adaptive and Neural Computationy, Edinburgh University, 1996. (www.anc.ed.ac.uk/ mjo/papers/intro.ps).

[4] Park, Y.R., Murray, T.J., and Chen, C. Prediction sun spots using a layered perception neural networks. *IEEE Transaction on Neural Networks*, 7(2):501–505, 1994.

Terminology

[1] Addrians, P. and Zanringe, D. *Data Mining*. Addison-Wesley, 2nd ed., 1996.

[2] Arminger, G. and Polozik, W. Data analysis and information systems, statistical and conceptual approaches. studies in classification data analysis and knowledge organization. In *Statistical models and neural networks* (Eds. O.E. Barndorf-Nielsen, J.L. Jensen, and W.S. Kendall), vol. 7, pages 243–260. Springer, 1996.

[3] Fausett, L. *Fundamental of Neural, Architecture, Algorithms and Applications*. Prentice Hall, 1994.

[4] Gueney, K. *An Introduction to Neural Networks*. University College London Press, 1997.

[5] Kasabov, N.K. *Foundations of Neural Networks. Fuzzy Systems and Knowledge Engineering*. MIT Press, 1996.

[6] Sarle, W.S. Neural network and statistical jargon. (ftp://ftp.sas.com).

[7] Sarle, W.S. Neural networks and statistical models. In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. 1994. (http//citeseer.nj.nec.com/sarle94neural.html).

[8] Stegeman, J.A. and Buenfeld, N.R. A glossary of basic neural network terminology for regression problems. *Neural Computing and Applications*, (8):290–296, 1999.

Some Common Neural Networks Models

Multilayer Feedforward Networks

[1] DeWilde, P. *Neural Network Models*. Springer, 2nd ed., 1997.

[2] Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd ed., 1999.

[3] Picton, P. *Neural Networks*. Palgrave, 2nd ed., 1999.

[4] Wu, C.H. and Mcharty, J.W. *Neural Networks and Genome Informatics*. Elsevier, 2000.

Associative and Hopfield Networks

[1] Abdi, H., Valentin, D., and Edelman, B. *SAGE, Quantitative Applications in the Social Sciences*. 124. Prentice Hall, 1999.

[2] Braga, A.P., Ludemir, T.B., and Carvalho, A.C.P.L.F. *Redes Neurais Artificiais: Teoria e Aplicações*. LTC editora, 2000.

[3] Chester, M. *Neural Networks: A Tutorial*. Prentice Hall, 1993.

[4] Fausett, L. *Fundamentals of Neural Networks Architectures, Algorithms and Applications*. Prentice Hall, 1994.

[5] Rojas, R. *Neural Networks: A Systematic Introduction*. Springer, 1996.

Radial Basis Function Networks

[1] Lowe, D. Statistics and neural networks advances in the interface. In *Radial basis function networks and statistics* (Eds. J.W. Kay and D.M.T. Herington), pages 65–95. Oxford University Press, 1999.

[2] Orr, M.J.L. Introduction to radial basis function networks. Tech. rep., Institute of Adaptive and Neural Computationy, Edinburgh University, 1996. (www.anc.ed.ac.uk/ mjo/papers/intro.ps).

[3] Wu, C.H. and McLarty, J.W. *Neural Networks and Genome Informatics*. Elsevier, 2000.

Wavelet Neural Networks

[1] Abramovich, F., Barley, T.C., and Sapatinas, T. Wavelet analysis and its statistical application. *Journal of the Royal Statistical Society D*, 49(1):1–29, 2000.

[2] Hubbard, B.B. *The World According to Wavelets*. A.K. Peters, LTD, 1998.

[3] Iyengar, S.S., Cho, E.C., and Phola, V.V. *Foundations of Wavelets Networks and Applications*. Chapman and Gall/CRC, 2002.

[4] Martin, E.B. and Morris, A.J. Statistics and neural networks. In *Artificial neural networks and multivariate statistics* (Eds. J.W. Kay and D.M. Titterington), pages 195–258. Oxford University Press, 1999.

[5] Morettin, P.A. From fourier to wavelet analysis of true series. In *Proceedings in Computational Statistics* (Ed. A. Prat), pages 111–122. Physica-Verlag, 1996.

[6] Morettin, P.A. Wavelets in statistics (text for a tutorial). In *The Third International Conference on Statistical Data Analysis Based on L_1-Norm and Related Methods*. Neuchâtel, Switzerland, 1997. (www.ime.usp/br/ pam/papers.html).

[7] Zhang, Q. and Benveniste, A. Wavelets networks. *IEEE Transactions on Neural Networks*, 3(6):889–898, 1992.

Mixture of Experts Networks

[1] Bishop, C.M. *Neural Network for Pattern Recognition*. 124. Oxford, 1997.

[2] Cheng, C. A neural network approach for the analysis of control charts patterns. *International J. of Production Research*, 35:667–697, 1997.

[3] Ciampi, A. and Lechevalier, Y. Statistical models as building blocks of neural networks. *Communications in Statistics - Theory and Methods*, 26(4):991–1009, 1997.

[4] Desai, V.S., Crook, J.N., and Overstreet, G.A. A comparison of neural networks and linear scoring models in the credit union environmental. *European Journal of Operational Research*, 95:24–37, 1996.

[5] Jordan, M.I. and Jacobs, R.A. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[6] Mehnotra, K., Mohan, C.K., and Ranka, S. *Elements of Artificial Neural Networks*. MIT Press, 2000.

[7] Ohno-Machodo, L., Walker, M.G., and Musen, M.A. Hierarchical neural networks for survival analysis. In *The Eighth Worlds Congress on Medical Information*. Vancouver, B.C., Canada, 1995.

Neural Network Models & Statistical Models Interface: Comments on the Literature

[1] Arminger, .G. and Enache, D. Statistical models and artificial neural networks. In *Data Analysis and Information Systems* (Eds. H.H. Bock and W. Polasek), pages 243–260. Springer Verlag, 1996.

[2] Balakrishnan, P.V.S., Cooper, M.C., Jacob, M., and Lewis, P.A. A study of the classification capabilities of neural networks using unsupervised learning: a comparision with k-means clustering. *Psychometrika*, (59):509–525, 1994.

[3] Bishop, C.M. *Neural Network for Pattern Recognition*. 124. Oxford, 1997.

[4] Cheng, B. and Titterington, D.M. Neural networks: A review from a statistical perspective (with discussion). *Statistical Sciences*, 9:2–54, 1994.

[5] Chryssolouriz, G., Lee, M., and Ramsey, A. Confidence interval predictions for neural networks models. *IEEE Transactions on Neural Networks*, 7:229–232, 1996.

[6] De Veaux, R.D., Schweinsberg, J., Schumi, J., and Ungar, L.H. Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282, 1998.

[7] Ennis, M., Hinton, G., Naylor, D., Revow, M., and Tibshirani, R. A comparison of statistical learning methods on the GUSTO database. *Statistics In Medicine*, 17:2501–2508, 1998.

[8] Hwang, J.T.G. and Ding, A.A. Prediction interval for artificial neural networks. *Journal of the American Statistical Association*, 92:748–757, 1997.

[9] Jordan, M.I. Why the logistic function? a tutorial discussion on probabilities and neural networks. Report 9503, MIT Computational Cognitive Science, 1995.

[10] Kuan, C.M. and White, H. Artificial neural networks: An econometric perspective. *Econometric Reviews*, 13:1–91, 1994.

[11] Lee, H.K.H. *Model Selection and Model Averaging for Neural Network*. Ph.d thesis., Cornegie Mellon University, Depatament of Statistics, 1998.

[12] Lee, H.K.H. *Bayesian Nonparametrics via Neural Networks*. SIAM: Society for Industrial and Applied Mathematics, 2004.

[13] Lee, T.H., White, H., and Granger, C.W.J. Testing for negleted nonlinearity in time series models. *Journal of Econometrics*, 56:269–290, 1993.

[14] Lowe, D. Statistics and neural networks advances in the interface. In *Radial basis function networks and statistics* (Eds. J.W. Kay and D.M.T. Herington), pages 65–95. Oxford University Press, 1999.

[15] Mackay, D.J.C. *Bayesian Methods for Adaptive Methods.* Ph.d thesis., California Institute of Technology, 1992.

[16] Mangiameli, P., Chen, S.K., and West, D. A comparison of SOM neural networks and hierarchical clustering methods. *European Journal of Operations Research*, (93):402–417, 1996.

[17] Martin, E.B. and Morris, A.J. Statistics and neural networks. In *Artificial neural networks and multivariate statistics* (Eds. J.W. Kay and D.M. Titterington), pages 195–258. Oxford University Press, 1999.

[18] Mingoti, S.A. and Lima, J.O. Comparing som neural network with fuzzy c-means, k-means and traditional hierarchical clustering algorithms. *European Journal of Operational Research*, 174(3):1742–1759, 2006.

[19] Murtagh, F. Neural networks and related “massively parallel” methods for statistics: A short overview. *International Statistical Review*, 62(3):275–288, october 1994.

[20] Neal, R.M. *Bayesian Learning for Neural Networks.* NY Springer, 1996.

[21] Paige, R.L. and Butler, R.W. Bayesian inference in neural networks. *Biometrika*, 88(3):623–641, 2001.

[22] Poli, I. and Jones, R.D. A neural net model for prediction. *Journal of the American Statistical Association*, 89(425):117–121, 1994.

[23] Rios-Insua, D. and Muller, P. Feedfoward neural networks for nonparametric regression. In *Pratical Nonparametric and Semiparametric Bayesian Statistics* (Eds. D. Dey, P. Mueller, and D. Sinha), pages 181–194. NY Springer, 1996.

[24] Ripley, B.D. *Pattern Recognition and Neural Networks.* Cambrige University Press, 1996.

[25] Sarle, W.S. Neural network and statistical jargon. (ftp://ftp.sas.com).

[26] Sarle, W.S. Neural networks and statistical models. In *Proceedings of the Nineteenth Annual SAS Users Group International Conference.* 1994. (http//citeseer.nj.nec.com/sarle94neural.html).

[27] Schwarzer, G., Vach, W., and Schumacher, M. On the misuses of artificial neural networks for prognostic and diagnostic classification in oncology. *Statistics in Medicine*, 19:541–561, 2000.

[28] Smith, M. *Neural Network for Statistical Modelling*. Van Nostrond Reinhold, 1993.

[29] Stern, H.S. Neural network in applied statistics (with discussion). *Technometrics*, 38:205–220, 1996.

[30] Swanson, N.R. and White, H. A model-selection approach to assessing the information in the term structure using linear models and artificial neural networks. *Journal of Business & Economic Statistics*, 13(3):265–75, 1995.

[31] Swanson, N.R. and White, H. Forecasting economic time series using flexible versus fixed specification and linear versus nonlinear econometric models. *International Journal of Forecasting*, 13(4):439–461, 1997.

[32] Tibshirani, R. A comparison of some error estimates for neural networks models. *Neural Computation*, 8:152–163, 1996.

[33] Titterington, D.M. Bayesian methods for neural networks and related models. *Statistical Science*, 19(1):128–139, 2004.

[34] Tu, J.V. Advantages and disadvantages of using artifical neural networks versus logit regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49:1225–1231, 1996.

[35] Vach, W., Robner, R., and Schumacher, M. Neural networks and logistic regression. part ii. *Computational Statistics and Data Analysis*, 21:683–701, 1996.

[36] Warner, B. and Misra, M. Understanding neural networks as statistical tools. *American Statistician*, 50:284–293, 1996.

[37] White, H. Neural-network learning and statistics. *AI Expert*, 4(12):48–52, 1989.

[38] White, H. Parametric statistical estimation using artificial neural networks. In *Mathematical Perspectives on Neural Networks* (Eds. P. Smolensky, M.C. Mozer, and D.E. Rumelhart), pages 719–775. L Erlbaum Associates, 1996.

[39] White, H. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1(4):425–464, 1989.

[40] White, H. Some asymptotic results for learning in single hidden-layer feedforward network models. *Journal of the American Statistical Association*, 84(408):1003–1013, 1989.

[41] Zhang, G.P. Avoiding pitfalls in neural network research. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 37(1):3–16, 2007.

Multivariate Statistics Neural Network Models

Cluster and Scaling Networks

[1] Balakrishnan, P.V.S., Cooper, M.C., Jacob, M., and Lewis, P.A. A study of the classification capabilities of neural networks using unsupervised learning: a comparision with k-means clustering. *Psychometrika*, (59):509–525, 1994.

[2] Braga, A.P., Ludemir, T., and Carvalho, . *Redes Neurais Artificias*. LTC Editora, Rio de Janeiro, 2000.

[3] Carvalho, J., Pereira, B.B., and Rao, C.R. Combining unsupervised and supervised neural networks analysis in gamma ray burst patterns. Techinical report center of multivariate analysis, Departament of Statistics, Penn State University, 2006.

[4] Chiu, C., Yeh, S.J., and Chen, C.H. Self-organizing arterial pressure pulse classification using neural networks: theoretical considerations and clinical applicability. *Computers in Biology and Medicine*, (30):71–78, 2000.

[5] Draghici, S. and Potter, R.B. Predicting hiv drug resistance with neural networks. *Bioinformatics*, (19):98–107, 2003.

[6] Fausett, L. *Fundamentals of Neural Networks Architectures, Algorithms and Applications*. Prentice Hall, 1994.

[7] Hsu, A.L., Tang, S.L., and Halgamuge, S.K. An unsupervised hierarchical dynamic self-organizing approach to cancer class discovery and marker gene identification in microarray data. *Bioinformatics*, 19:2131–2140, 2003.

[8] Ishihara, S., Ishihara, K., Nagamashi, M., and Matsubara, Y. An automatic builder for a kansei engineering expert systems using a self-organizing neural networks. *Intenational Journal of Industrial Ergonomics*, (15):13–24, 1995.

[9] Kartalopulos, S.V. *Understanding Neural Networks and Fuzzy Logic. Basic Concepts and Applications*. IEEE Press, 1996.

[10] Lourenço, P.M. *Um Modelo de Previsão de Curto Prazo de Carga Elétrica Combinando Métodos Estatísticos e Inteligência Computacional*. D. sc. in eletrical engineering, Pontifícia Universidade Católica do Rio de Janeiro, 1998.

[11] Maksimovic, R. and Popovic, M. Classification of tetraplegies through automatic movement evaluation. *Medical Engineering and Physics*, (21):313–327, 1999.

[12] Mangiameli, P., Chen, S.K., and West, D. A comparison of SOM neural networks and hierarchical clustering methods. *European Journal of Operations Research*, (93):402–417, 1996.

[13] Mehrotra, K., Mohan, C.K., and Ranka, S. *Elements of Artificial Neural Networks*. MIT Press, 2000.

[14] Patazopoulos, D., P, P.K., Pauliakis, A., Iokim-liossi, A., and Dimopoulos, M.A. Static cytometry and neural networks in the discrimination of lower urinary system lesions. *Urology*, (6):946–950, 1988.

[15] Rozenthal, M. *Esquizofrenia: Correlação Entre Achados Neuropsicológicos e Sintomatologia Através de Redes Neuronais Artificiais*. D.Sc. in Psychiatry, UFRJ, 2001.

[16] Rozenthal, M., Engelhart, E., and Carvalho, L.A.V. Adaptive resonance theory in the search for neuropsychological patterns of schizophrenia. Techinical Report in Systems and Computer Sciences, COPPE - UFRJ, May 1998.

[17] Santos, A.M. *Redes neurais e Árvores de Classificação Aplicados ao Diagnóstico da Tuberculose Pulmonar Paucibacilar*. D. Sc. in Production Engineering, COPPE - UFRJ, 2003.

[18] Vuckovic, A., Radivojevic, V., Chen, A.C.N., and Popovic, D. Automatic recognition of aletness and drawsiness from eec by an artificial neural networks. *Medical Engineering and Physics*, (24):349–360, 2002.

Dimensional Reduction Networks

[1] Cha, S.M. and Chan, L.E. Applying independent component analysis to factor model in finance. In *Intelligent Data Engineering and Automated Learning, IDEAL 2000, Data Mining, Finacncial Enginerring and Inteligent Agents* (Eds. K. Leung, L.W. Chan, and H. Meng), pages 538–544. Springer, Bruges, april 2002.

[2] Chan, L.W. and Cha, S.M. Selection of independent factor model in finance. In *Proceeding of the 3th International Conference on Independent Component Analyses* (Eds. K. Leung, L.W. Chan, and H. Meng), pages 161–166. december 2001.

[3] Clausen, S.E. Applied correspondence analysis: An introduction. *SAGE, Quantitative Applications in the Social Sciences*, (121), 1998.

[4] Comon, P. Independent component analysis, a new concept? *Signal Processing*, 3:287–314, 1994.

[5] Correia-Perpinan, M.A. A review of dimensional reduction techniques. Techinical report cs-96-09, University of Sheffield, 1997.

[6] Darmois, G. Analise générale des liasons stochastiques. *International Statistical Review*, 21:2–8, 1953.

[7] Delichere, M. and Memmi, D. Neural dimensionality reduction for document processing. In *European Sympsium on Artificial Neural Networks - ESANN*. april 2002a.

[8] Delichere, M. and Memmi, D. Analyse factorielle neuronalle pour documents textuels. In *Traiment Automatique des Languages Naturelles - TALN*. june 2002b.

[9] Diamantaras, K.I. and Kung, S.Y. *Principal Components Neural Networks. Theory and Applications*. Wiley, 1996.

[10] Draghich, S., Graziano, F., Kettoola, S., Sethi, I., and Towfic, G. Mining HIV dynamics using independent component analysis. *Bioinformatics*, 19:931–986, 2003.

[11] Duda, R.O., Hart, P.E., and Stock, D.G. *Pattern Classification*. Wiley, 2nd ed., 2001.

[12] Dunteman, G.H. Principal component analysis. *SAGE,Quantitative Applications in the Social Sciences*, (69), 1989.

[13] Fiori, S. Overview of independent component analysis technique with application to synthetic aperture radar (SAR) imagery processing. *Neural Networks*, 16:453–467, 2003.

[14] Giannakopoulos, X., Karhunen, J., and Oja, E. An experimental comparision of neural algorithms for independent component analysis and blind separation. *International Journal of Neural Systems*, 9:99–114, 1999.

[15] Girolami, M. *Self-Organizing Neural Networks: Independent Component Analysis and Blind Source Separation*. Springer, 1999.

[16] Gnandesikan, R. *Methods of Statistical Data Analysis of Multivariate Observations*. Wiley, 2nd ed., 1991.

[17] Hertz, J., Krogh, A., and Palmer, R.G. *Introduction to the Theory of Neural Computation*. Addison-Westey, 1991.

[18] Hyvarinen, A. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[19] Hyvarinen, A. and Oja, E. Independent component analysis: algorithm and applications. *Neural Networks*, 13:411–430, 2000.

[20] Hyvärinen, A., Karhunen, J., and Oja, E. *Independent Component Analysis*. Wiley, 2001.

[21] Jackson, J.E. *A User´s Guide to Principal Components*. Wiley, 1991.

[22] Jutten, C. and Taleb, A. Source separation: from dust till down. In *Procedings ICA 2000*. 2000.

[23] Kagan, A.M., Linnik, Y., and Rao, C.R. *Characterization Problems in Mathematical Statistics*. Wiley, 1973.

[24] Karhunen, J., Oja, E., Wang, L., and Joutsensalo, J. A class of neural networks for independent component analysis. *IEEE Transactions on Neural Networks*, 8:486–504, 1997.

[25] Kin, J.O. and Mueller, C.W. Introduction to factor analysis. *SAGE, Quantitative Applications in the Social Sciences*, (13), 1978.

[26] Lebart, L. Correspondence analysis and neural networks. In *IV Internation Meeting in Multidimensional Data Analysis - NGUS-97*, pages 47–58. K. Fernandes Aquirre (Ed), 1997.

[27] Lee, T.W. *Independent Component Analysis Theory and Applications*. Kluwer Academic Publishers, 1998.

[28] Lee, T.W., Girolami, M., and Sejnowski, T.J. Independent component analysis using an extended informax algorithm for mixed subgaussian and supergaussian sources. *Neural Computation*, 11:417–441, 1999.

[29] Mehrotra, K., Mohan, C.K., and Ranka, S. *Elements of Artificial Neural Networks*. MIT Press, 2001.

[30] Nicole, S. Feedfoward neural networks for principal components extraction. *Computational Statistics and Data Analysis*, 33:425–437, 2000.

[31] Pielou, E.C. *The Interpretation of Ecological Data: A primer on Classification and Ordination*. Willey, 1984.

[32] Ripley, B.D. *Pattern Recognition and Neural Networks*. Cambrige University Press, 1996.

[33] Rowe, D.B. *Multivariate Bayesian Statistics Models for Source Separation and Signal Unmixing*. Chapman & Hall/CRC, 2003.

[34] Stone, J.V. and Porril, J. Independent component analysis and projection pursuit: a tutorial introduction. Techinical Report, Psychology Departament, Sheffield University, 1998.

[35] Sun, K.T. and Lai, Y.S. Applying neural netorks technologies to factor analysis. In *Procedings of the National Science Council, ROD(D)*, vol. 12, pages 19–30. K. Fernandes Aquirre (Ed), 2002.

[36] Tura, B.R. *Aplicação do Data Mining em Medicina*. M. sc. thesis, Faculdade de Medicina, Universidade Federal do Rio de Janeiro, 2001.

[37] Vatentin, J.L. *Ecologia Numérica: Uma Introdução à Análise Multivariada de Dados Ecológicos*. Editora Interciência, 2000.

[38] Weller, S. and Rommey, A.K. Metric scalling: Correspondence analysis. *SAGE Quantitative Applications in the Social Sciences*, (75), 1990.

Classification Networks

[1] Arminger, G., Enache, D., and Bonne, T. Analyzing credit risk data: a comparision of logistic discrimination, classification, tree analysis and feedfoward networks. *Computational Statistics*, 12:293–310, 1997.

[2] Asparoukhov, O.K. and Krzanowski, W.J. A comparison of discriminant procedures for binary variables. *Computational Statistics and Data Analysis*, 38:139–160, 2001.

[3] Bose, S. Multilayer statistical classifiers. *Computational Statistics and Data Analysis*, 42:685–701, 2003.

[4] Bounds, D.G., Lloyd, P.J., and Mathew, B.G. A comparision of neural network and other pattern recognition aproaches to the diagnosis of low back disorders. *Neural Networks*, 3:583–591, 1990.

[5] Desai, V.S., Crook, J.N., and Overstreet, G.A. A comparision of neural networks and linear scoring models in the credit union environment. *European Journal of Operational Research*, 95:24–37, 1996.

[6] Fisher, R.A. The use of multiples measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[7] Lippmann, R.P. Neural nets for computing acoustic speech and signal processing. In *ICASSP*, vol. 1, pages 1–6. 1988.

[8] Lippmann, R.P. Pattern classification using neural networks. *IEEE Comunications Magazine*, 11:47–64, 1989.

[9] Lippmann, R.P. A critical overview of neural networks pattern classifiers. *Neural Networks for Signal Processing*, 30:266–275, 1991.

[10] Markham, I.S. and Ragsdale, C.T. Combining neural networks and statistical predictors to solve the classification problem in discriminant analysis. *Decision Sciences*, 26:229–242, 1995.

[11] Randys, S. *Statistical and Neural Classifiers*. Springer, 1949.

[12] Rao, C.R. On some problems arising out of discrimination with multiple characters. *Sankya*, 9:343–365, 1949.

[13] Ripley, B.D. Neural networks and related methods for classification (with discussion). *Journal of the Royal Statistical Society B*, 56:409–456, 1994.

[14] Santos, A.M. *Redes Neurais e Árvores de Classificação Aplicadas ao Diagnóstico de Tuberculose Pulmonar*. Tese D.Sc. Engenharia de Produção, UFRJ, 2003.

[15] Santos, A.M., Seixas, J.M., Pereira, B.B., Medronho, R.A., Campos, M.R., and Calôba, L.P. Usando redes neurais artificiais e regressão logística na predição de hepatite A. *Revista Brasileira de Epidemologia*, (8):117–126, 2005.

[16] Santos, A.M., Seixas, J.M., Pereira, B.B., Mello, F.C.Q., and Kritski, A. Neural networks: an application for predicting smear negative pulmonary tuberculosis. In *Advances in Statistical Methods for the Health Sciences: Applications to Cancer and AIDS Studies, Genome Sequence and Survival Analysis* (Eds. N. Balakrishman, J.H. Auget, M. Mesbab, and G. Molenberghs), pages 279–292. Springer, 2006.

[17] West, D. Neural networks credit scoring models. *Computers and Operations Research*, 27:1131–1152, 2000.

Regression Neural Network Models

Generalized Linear Model Networks

[1] Nelder, J.A. and McCullagh, P. *Generalized Linear Model*. CRC/Chapman and Hall, 2 nd ed., 1989.

[2] Sarle, W.S. Neural networks and statistical models. In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. 1994. (http//citeseer.nj.nec.com/sarle94neural.html).

[3] Warner, B. and Misra, M. Understanding neural networks as statistical tools. *American Statistician*, 50:284–293, 1996.

Logistic Regression Networks

[1] Arminger, G., Enache, D., and Bonne, T. Analyzing credit risk data: a comparision of logistic discrimination, classification, tree analysis and feedfoward networks. *Computational Statistics*, 12:293–310, 1997.

[2] Boll, D., Geomini, P.A.M.J., Brölmann, H.A.M., Sijmons, E.A., Heintz, P.M., and Mol, B.W.J. The pre-operative assessment of the adnexall mass: The accuracy of clinical estimates versus clinical prediction rules. *International Journal of Obstetrics and Gynecology*, 110:519–523, 2003.

[3] Brockett, P.H., Cooper, W.W., Golden, L.L., and Xia, X. A case study in applying neural networks to predicting insolvency for property and causality insurers. *Journal of the Operational Research Society*, 48:1153–1162, 1997.

[4] Carvalho, M.C.M., Dougherty, M.S., Fowkes, A.S., and Wardman, M.R. Forecasting travel demand: A comparison of logit and artificial neural network methods. *Journal of the Operational Research Society*, 49:717–722, 1998.

[5] Ciampi, A. and Zhang, F. A new approach to training back-propagation artificial neural networks: Empirical evaluation on ten data sets from clinical studies. *Statistics in Medicine*, 21:1309–1330, 2002.

[6] Cooper, J.C.B. Artificial neural networks versus multivariate statistics: An application from economics. *Journal of Applied Statistics*, 26:909–921, 1999.

[7] Faraggi, D. and Simon, R. The maximum likelihood neural network as a statistical classification model. *Journal of Statistical Planning and Inference*, 46:93–104, 1995.

[8] Fletcher, D. and Goss, E. Forecasting with neural networks: An application using bankruptcy data. *Information and Management*, 24:159–167, 1993.

[9] Hammad, T.A., Abdel-Wahab, M.F., DeClaris, N., El-Sahly, A., El-Kady, N., and Strickland, G.T. Comparative evaluation of the use of neural networks for modeling the epidemiology of schistosomosis mansoni. *Transaction of the Royal Society of Tropical Medicine and Hygiene*, 90:372–376, 1996.

[10] Lenard, M.J., Alam, P., and Madey, G.R. The application of neural networks and a qualitative response model to the auditor's going concern uncertainty decision. *Decision Sciences*, 26:209–226, 1995.

[11] Nguyen, T., Malley, R., Inkelis, S.K., and Kupperman, N. Comparison of prediction models for adverse outcome in pediatric meningococcal disease using artificial neural network and logistic regression analysis. *Journal of Clinical Epidemiology*, 55:687–695, 2002.

[12] Ottenbacher, K.J., Smith, P.M., Illig, S.B., Limm, E.T., Fiedler, R.C., and Granger, C.V. Comparison of logistic regression and neural networks to predict rehospitalization in patients with stroke. *Journal of Clinical Epidemiology*, 54:1159–1165, 2001.

[13] Schumacher, M., Roßner, R., and Vach, W. Neural networks and logistic regression. part i. *Computational Statistics and Data Analysis*, 21:661–682, 1996.

[14] Schwarzer, G., Vach, W., and Schumacher, M. On the misuses of artificial neural networks for prognostic and diagnostic classification in oncology. *Statistics in Medicine*, 19:541–561, 2000.

[15] Tu, J.V. Advantages and disadvantages of using artifical neural networks versus logit regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49:1225–1231, 1996.

[16] Vach, W., Robner, R., and Schumacher, M. Neural networks and logistic regression. part ii. *Computational Statistics and Data Analysis*, 21:683–701, 1996.

Regression Networks

[1] Balkin, S.D.J. and Lim, D.K.J. A neural network approach to response surface methodology. *Communications in Statistics - Theory and Methods*, 29(9/10):2215–2227, 2000.

[2] Church, K.B. and Curram, S.P. Forecasting consumer's expenditure: A comparison between econometric and neural networks models. *International Journal of Forecasting*, 12:255–267, 1996.

[3] Gorr, W.L., Nagin, D., and Szczypula, J. Comparative study of artificial neural networks and statistical models for predicting student grade point average. *International Journal of Forecasting*, 10:17–34, 1994.

[4] Laper, R.J.A., Dalton, K.J., Prager, R.W., Forsstrom, J.J., Selbmann, H.K., and Derom, R. Application of neural networks to the ranking of perinatal variables influencing birthweight. *Scandinavian Journal of Clinical and Laboratory Investigation*, 55 (suppl 22):83–93, 1995.

[5] Qi, M. and Maddala, G.S. Economic factors and the stock market: A new perspective. *Journal of Forecasting*, 18:151–166, 1999.

[6] Silva, A.B.M., Portugal, M.S., and Cechim, A.L. Redes neurais artificiais e analise de sensibilidade: Uma aplicação à demanda de importações brasileiras. *Economia Aplicada*, 5:645–693, 2001.

[7] Tkacz, G. Neural network forecasting of canadian gdp growth. *International Journal of Forecasting*, 17:57–69, 2002.

Nonparametric Regression and Classification Networks

[1] Ciampi, A. and Lechevalier, Y. Statistical models as building blocks of neural networks. *Communications in Statistics Theory and Methods*, 26(4):991–1009, 1997.

[2] Fox, J. Nonparametric simple regression. *Quantitative Applications in the Social Sciences*, 130, 2000a.

[3] Fox, J. Multiple and generalized nonparametric regression. *Quantitative Applications in the Social Sciences*, 131, 2000b.

[4] Kim, S.H. and Chun, S.H. Grading forecasting using an array of bipolar predictions: Application of probabilistic networks to a stock market index. *International Journal of Forecasting*, 14:323–337, 1998.

[5] Peng, F., Jacobs, R.A., and Tanner, M.A. Bayesian inference in mixture-of-experts and hierarchical mixtures-of-experts models with an application to speech recognition. *Journal of the American Statistical Association*, 91:953–960, 1996.

[6] Picton, P. *Neural Networks*. PARLGRAVE Ed., 2 nd ed., 2000.

[7] Ripley, B.D. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[8] Specht, D.F. A general regression neural networks. *IEEE Transactions on Neural Networks*, 2(6):568–576, 1991.

[9] Specht, D.F. Probabilistic neural networks for classification, mapping, or associative memories. In *Proceedings International Conference on Neural Networks*, vol. 1, pages 525–532. 1998.

Survival Analysis, Time Series and Control Chart Neural Network Models and Inference

Survival Analysis Networks

[1] Bakker, B., Heskes, T., and Kappen, B. Improving lox survival analysis with a neural bayesian approach in medicine, 2003. (in press).

[2] Biganzoli, E., Baracchi, P., and Marubini, E. Feed-forward neural networks for the analysis of censored survival data: A partial logistic regression approach. *Statistics in Medicine*, 17:1169–1186, 1998.

[3] Biganzoli, E., Baracchi, P., and Marubini, E. A general framework for neural network models on censored survival data. *Neural Networks*, 15:209–218, 2002.

[4] Ciampi, A. and Etezadi-Amoli, J. A general model for testing the proportional harzards and the accelerated failure time hypothesis in the analysis of censored survival data with covariates. *Communications in Statistics A*, 14:651–667, 1985.

[5] Faraggi, D., LeBlanc, M., and Crowley, J. Understanding neural networks using regression trees: An application to multiply myeloma survival data. *Statistics in Medicine*, 20:2965–2976, 2001.

[6] Faraggi, D. and Simon, R. A neural network model for survival data. *Statistics in Medicine*, 14:73–82, 1995a.

[7] Faraggi, D. and Simon, R. The maximum likelihood neural network as a statistical classification model. *Journal of Statistical Planning and Inference*, 46:93–104, 1995b.

[8] Faraggi, D., Simon, R., Yaskily, E., and Kramar, A. Bayesian neural network models for censored data. *Biometrical Journal*, 39:519–532, 1997.

[9] Groves, D.J., Smye, S.W., Kinsey, S.E., Richards, S.M., Chessells, J.M., Eden, O.B., and Basley, C.C. A comparison of cox regression and neural networks for risk stratification in cases of acute lymphoblastic leukemia in children. *Neural Computing and Applications*, 8:257–264, 1999.

[10] Kalbfleish, J.D. and Prentice, R.L. *The Statistical Analysis of Failure Data*. Wisley, 2 nd ed., 2002.

[11] Lapuerta, P., Azen, S.P., and LaBree, L. Use of neural networks in predicting the risk of coronary arthery disease. *Computer and Biomedical Research*, 28:38–52, 1995.

[12] Liestol, K., Andersen, P.K., and Andersen, U. Survival analysis and neural nets. *Statistics in Medicine*, 13:1189–1200, 1994.

[13] Louzada-Neto, F. Extended hazard regression model for rehability and survival analysis. *Lifetime Data Analysis*, 3:367–381, 1997.

[14] Louzada-Neto, F. Modeling life data: A graphical approach. *Applied Stochastic Models in Business and Industry*, 15:123–129, 1999.

[15] Louzada-Neto, F. and Pereira, B.B. Modelos em análise de sobrevivência. *Cadernos Saúde Coletiva*, 8(1):9–26, 2000.

[16] Mariani, L., Coradin, D., Bigangoli, E., Boracchi, P., Marubini, E., Pillot, S., Salvadori, R., Verones, V., Zucali, R., and Rilke, F. Prognostic factors for metachronous contralateral breast cancer: a comparision of the linear Cox regression model and its artificial neural networks extension. *Breast Cancer Research and Threament*, 44:167–178, 1997.

[17] Ohno-Machado, L. A comparison of cox proportional hazard and artificial network models for medical prognoses. *Computers in Biology and Medicine*, 27:55–65, 1997.

[18] Ohno-Machado, L., Walker, M.G., and Musen, M.A. Hierarchical neural networks for survival analysis. In *The Eighth World Congress on Medical Information*. Vancouver, BC, Canada, 1995.

[19] Ripley, B.D. and Ripley, R.M. Artificial neural networks: Prospects for medicine. In *Neural networks as statistical methods in survival analysis* (Eds. R. Dybowski and C. Gant). Landes Biosciences Publishing, 1998.

[20] Ripley, R.M. *Neural network models for breast cancer prognoses. D. Phill. Thesis*. Department of Engineering Science, University of Oxford, 1998. (www.stats.ox.ac.uk/ ruth/index.html).

[21] Xiang, A., Lapuerta, P., Ryutov, Buckley, J., and Azen, S. Comparison of the performance of neural networks methods and cox regression for censored survival data. *Computational Statistics and Data Analysis*, 34:243–257, 2000.

Time Series Forecasting Networks

[1] Balkin, S.D. and Ord, J.K. Automatic neural network modeling for univariate time series. *International Journal of Forecasting*, 16:509–515, 2000.

[2] Caloba, G.M., Caloba, L.P., and Saliby, E. Cooperação entre redes neurais artificiais e técnicas classicas para previsão de demanda de uma série de cerveja na australia. *Pesquisa Operacional*, 22:347–358, 2002.

[3] Chakraborthy, K., Mehrotra, K., Mohan, C.K., and Ranka, S. Forecasting the behavior of multivariate time series using neural networks. *Neural Networks*, 5:961–970, 1992.

[4] Cramer, H. On the representation of functions by certain fourier integrals. *Transactions of the American Math. Society*, 46:191–201, 1939.

[5] Donaldson, R.G. and Kamstra, M. Forecasting combining with neural networks. *Journal of Forecasting*, 15:49–61, 1996.

[6] Faraway, J. and Chatfield, C. Time series forecasting with neural networks: A comparative study using the airline data. *Applied Statistics*, 47:231–250, 1998.

[7] Ghiassi, M., Saidane, H., and Zimbra, D.K. A dynamic artificial neural network for forecasting series events. *International Journal of Forecasting*, 21:341–362, 2005.

[8] Ghiassi, M., Saidane, H., and Zimbra, D.K. A dynamic artificial neural network model for forecasting series events. *International Journal of Forecasting*, 21:341–362, 2005.

[9] Hill, T., Marquez, L., O'Connor, M., and Remus, W. Artificial neural network models for forecasting and decision making. *International Journal of Forecasting*, 10:5–15, 1994.

[10] Kajitomi, Y., Hipel, K.W., and Mclead, A.I. Forecasting nonlinear time series with feed-foward neural networks: a case of study of canadian lynx data. *Journal of Forecasting*, 24:105–117, 2005.

[11] Kajitoni, Y., Hipel, K.W., and Mclead, A.I. Forecasting nonlinear time series with feed-foward neural networks: a case study of canadian lynx data. *Journal of Forecasting*, 24:105–117, 2005.

[12] Lai, T.L. and Wong, S.P.S. Stochastic neural networks with applications to nonlinear time series. *Journal of the American Statistical Association*, 96:968–981, 2001.

[13] Li, X., Ang, C.L., and Gray, R. Stochastic neural networks with applications to nonlinear time series. *Journal of Forecasting*, 18:181–204, 1999.

[14] Lourenco, P.M. *Um Modelo de Previsão de Curto Prazo de Carga Elétrica Combinando Métodos de Estatística e Inteligência Computacional.* D.Sc Thesis, PUC-RJ, 1998.

[15] Machado, M.A.S., Souza, R.C., Caldeva, A.M., and Braga, M.J.F. Previsão de energia elétrica usando redes neurais nebulosas. *Pesquisa Naval*, 15:99–108, 2002.

[16] Park, Y.R., Murray, T.J., and Chen, C. Prediction sun spots using a layered perception *IEEE Transactions on Neural Networks*, 7(2):501–505, 1994.

[17] Pereira, B.B. Series temporais multivariadas. *Lectures Notes of SINAPE*, 15:99–108, 1984.

[18] Pino, R., de la Fuente, D., Parreno, J., and Pviore, P. Aplicacion de redes neuronales artificiales a la prevision de series temporales no estacionarias a no invertibles. *Qüestiio*, 26:461–482, 2002.

[19] Poli, I. and Jones, R.D. A neural net model for prediction. *Journal of the American Statistical Association*, 89(425):117–121, 1994.

[20] Portugal, M.S. Neural networks versus true series methods: A forecasting exercise. *Revista Brasileira de Economia*, 49:611–629, 1995.

[21] Raposo, C.M. *Redes Neurais na Previsão em Séries Temporais*. D.Sc Thesis, COPPE/UFRJ, 1992.

[22] Shamseldin, D.Y. and O'Connor, K.M. A non-linear neural network technique for updating of river flow forecasts. *Hydrology and Earth System Sciences*, 5:577–597, 2001.

[23] Stern, H.S. Neural network in applied statistics (with discussion). *Technometrics*, 38:205–220, 1996.

[24] Stern, H.S. Neural network in applied statistics (with discussion). *Technometrics*, 38:205–220, 1996.

[25] Terasvirta, T., Dijk, D.V., and Medeiros, M.C. Linear models, smooth transistions autoregressions and neural networks for forecasting macroeconomic time series: a re-examination. *International Journal of Forecasting*, 21:755–774, 2005.

[26] Terasvirta, T., Dijk, D.V., and Medeiros, M.C. Linear models, smooth transitions autoregressions and neural networks for forecasting macroeconomic time series: a re-examination. *Internation Journal of Forecasting*, 21:755–774, 2005.

[27] Tseng, F.M., Yu, H.C., and Tzeng, G.H. Combining neural network model with seasonal time series. *ARIMA model Technological Forecasting and S... Change*, 69:71–87, 2002.

[28] Venkatachalam, A.R. and Sohl, J.E. An intelligent model selection and forecasting system. *Journal of Forecasting*, 18:167–180, 1999.

[29] Zhang, G., Patuwo, B.E., and Hu, M.Y. Forecasting with artificial neural networks: the state of the art. *International Journal of Forecasting*, 14:35–62, 1998.

Control Chart Networks

[1] Cheng, C.S. A neural network approach for the analysis of control charts patterns. *International Journal of Production Research*, 35:667–697, 1997.

[2] Hwarng, H.B. Detecting mean shift in AR(1) process. *Decision Sciences Institute 2002 Annual Meeting Proceedings*, pages 2395–2400, 2002.

[3] Prybutok, V.R., Sanford, C.C., and Nam, K.T. Comparison of neural network to Shewhart $\bar{X}$ control chart applications. *Economic Quality Control*, 9:143–164, 1994.

[4] Smith, A.E. X-bar and R control chart using neural computing. *International Journal of Production Research*, 32:277–286, 1994.

Some Statistical Inference Results

Estimation Methods

[1] Aitkin, M. and Foxall, R. Statistical modelling of artificial neural networks using the multilayer perceptron. *Statistics and Computing*, 13:227–239, 2003.

[2] Belue, L.M. and Bauer, K.W. Designing experiments for single output multilayer perceptrons. *Journal of Statistical Computation and Simulation*, 64:301–332, 1999.

[3] Copobianco, E. Neural networks and statistical inference: seeking robust and efficient learning. *Journal of Statistics and Data Analysis*, 32:443–445, 2000.

[4] Faraggi, D. and Simon, R. The maximum likelihood neural network as a statistical classification model. *Journal of Statistical Planning and Inference*, 46:93–104, 1995.

[5] Lee, H.K.H. *Model Selection and Model Averaging for Neural Network*. Ph.d thesis., Cornegie Mellon University, Depatament of Statistics, 1998.

[6] Mackay, D.J.C. *Bayesian Methods for Adaptive Methods*. Ph.d thesis., California Institute of Technology, 1992.

[7] Neal, R.M. *Bayesian Learning for Neural Networks*. NY Springer, 1996.

[8] Rios-Insua, D. and Muller, P. Feedfoward neural networks for nonparametric regression. In *Pratical Nonparametric and Semiparametric Bayesian Statistics* (Eds. D. Dey, P. Mueller, and D. Sinha), pages 181–194. NY Springer, 1996.

[9] Schumacher, M., Robner, and Varch, W. Neural networks and logistic regression: Part i. *Computational Statistics and Data Analysis*, 21:661–682, 1996.

Interval Estimation

[1] Chryssolouriz, G., Lee, M., and Ramsey, A. Confidence interval predictions for neural networks models. *IEEE Transactions on Neural Networks*, 7:229–232, 1996.

[2] De Veaux, R.D., Schweinsberg, J., Schumi, J., and Ungar, L.H. Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282, 1998.

[3] Hwang, J.T.G. and Ding, A.A. Prediction interval for artificial neural networks. *Journal of the American Statistical Association*, 92:748–757, 1997.

[4] Tibshirani, R. A comparison of some error estimates for neural networks models. *Neural Computation*, 8:152–163, 1996.

Statistical Test

[1] Adams, J.B. Predicting pickle harvests using parametric feedfoward network. *Journal of Applied Statistics*, 26:165–176, 1999.

[2] Blake, A.P. and Kapetanious, G. A radial basis function artificial neural network test for negleted. *The Econometrics Journal*, 6:357–373, 2003.

[3] Kiani, K. On performance of neural networks nonlinearity tests: Evidence from G5 countries. *Departament of Economics - Kansas State University*, 2003.

[4] Lee, T.H. Neural network test and nonparametric kernel test for negleted nonlinearity in regression models. *Studies in Nonlinear Dynamics and Econometrics*, v4 issue 4, Article 1, 2001.

[5] Lee, T.H., White, H., and Granger, C.W.J. Testing for negleted nonlinearity in time series models. *Journal of Econometrics*, 56:269–290, 1993.

[6] Terasvirta, T., Lin, C.F., and Granger, C.W.J. Power properties of the neural network linearity test. *Journal of Time Series Analysis*, 14:209–220, 1993.